Clemson University
TigerPrints

All Theses | Theses

12-2015

# Parking lot monitoring system using an autonomous quadrotor UAV

Venkataraman Ganesh
*Clemson University*, venkatg@clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

 Part of the Computer Engineering Commons

Recommended Citation
Ganesh, Venkataraman, "Parking lot monitoring system using an autonomous quadrotor UAV" (2015). *All Theses*. 2293.
https://tigerprints.clemson.edu/all_theses/2293

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

# Parking lot monitoring system using an Autonomous quadrotor UAV

---

A Dissertation
Presented to
the Graduate School at
Clemson University

---

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Venkataraman Ganesh
December 2015

---

Accepted by:
Dr. Richard E. Groff, Committee Chair
Dr. Ian D. Walker
Dr. Timothy C. Burg

# Abstract

This work aims to investigate the use of a drone-based system to recognize license plates of vehicles in a parking lot. Many parking lots contain surveillance cameras mounted on walls or light towers for indoor and outdoor lots respectively. Parked vehicles are commonly monitored by law enforcement agents by driving around the parking lot to identify license plates with an onboard camera or a handheld device. However, these systems use expensive hardware and proprietary software. The main goal of this thesis is to develop an autonomous parking lot surveillance system using low-cost hardware and open-source software. Similar to wall-mounted surveillance cameras, a drone-based system can monitor parking lots without affecting the flow of traffic while also offering the mobility of patrol vehicles.

The Parrot AR Drone 2.0 is the quadrotor drone used in this work due to its modularity and cost efficiency. Video and navigation data (including GPS) are communicated to a host computer using a Wi-Fi connection. The host computer analyzes navigation data using a custom flight control loop to determine control commands to be sent to the drone. A new license plate recognition pipeline is used to identify license plates of vehicles from inflight video footage.

Although license plate recognition is a well studied topic, previous academic works have exploited discernibility between characters in high resolution images obtained from stationary cameras. In this work, the motion of the camera presents a novel challenge to the task of license plate recognition.

# Dedication

*As the bow bends, so the arrow flies. Parents are the bow whence children launch.*

*-Khalil Gibran*

I dedicate this work to my parents whose love, hard work and sacrifice have led me this far.

# Acknowledgments

I would like to thank my adviser Dr.Richard Groff for all the cheerful and constructive meeting sessions. This work would not have been possible without his guidance.

I would also like to express my gratitude to my first adviser Dr.Tim Burg for ensuring my successful transition between projects and to Dr.Ian Walker for agreeing to serve on my advisory committee. I would also like to acknowledge Dr.Stan Birchfield whose Image Processing course introduced me to the beautiful world of computer vision.

I am deeply indebted to Reinaldo Gift, my good friend and partner on the drone project, for his assistance in system development and experimentation.

I would like to extend my sincerest thanks and appreciation to the following people for their role in the successful completion of this thesis:

- My father Ganesh, mother Usha, and my sister Sneha, for their eternal love and support.
- Riddhi Naik, for bringing romance and craziness into my life.
- My friends Venkatramanan CM, Raj Navalakha and Siddharth Vedullapalli for providing a home away from home.
- Vivek Raman and Abhishek Pandey for their worldly advice and encouragement.
- The department of Electrical and Computer engineering at Clemson University, for giving me several valuable opportunities and for funding my education and research.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The number of motor vehicles on the road today is estimated to be over 1.5 billion, with some observers projecting 2 billion by 2035 [1]. An increase in the number of vehicles will inevitably lead to an increase in demand for parking lots. Vehicles and parking lots need to be monitored for occupancy, security and compliance. This has resulted in a great deal of interest in vehicle monitoring systems. Several systems have been developed for parking lot monitoring, most of which use stationery cameras mounted on walls and on light towers for monitoring indoor and outdoor parking spaces respectively.

Another category of vehicle monitoring systems called the License plate (LP) recognition systems were developed for identifying vehicle license plates at automated toll gates and parking lot gates. State-of-the-art LP recognition systems mounted on patrol vehicles allow real time recognition of license plates of both moving and parked vehicles. License plate recognition has been widely researched in both academia and industry, undergoing continuous improvement with the evolution of computer vision algorithms and embedded system hardware.

The advent of programmable Unmanned Aerial Vehicles (UAVs) has given rise to a great deal of interest in applications using UAVs. The Parrot AR Drone, DJI Phantom and the 3D Robotics quadrotor drones are all reasonably priced and ship with a host of onboard

sensors such as cameras and GPS as well as customizable flight control software. These UAVs have been used for terrain mapping, crop monitoring and photography [2]. Amazon Inc. has recently investigated the use of UAVs for delivering small packages [3].

This work is based on the idea of using an autonomous quadrotor drone to identify license plates of vehicles in a parking lot. The main motivation for this thesis is to find cost-effective solution to the parking lot surveillance and vehicle identification problem and to eliminate the need for fuel-based systems to monitor parking lots.

There are two goals to this thesis. Firstly, a quadrotor drone will be used to develop a new parking lot monitoring system. The drone will be programmed to fly along a user-defined path using GPS. The drone will communicate video and navigation data with a host system through Wi-Fi. Secondly, a new computer vision pipeline will be developed to perform license plate recognition on incoming video from the drone.

The parking lot monitoring system developed in this work uses the Parrot AR Drone 2.0 quadrotor drone because of its modularity and cost efficiency. The AR Drone Software Development kit provides the application programming interface to develop custom flight plans using GPS and other sensors. A new Python application called Fly-Py was developed to easily retrieve the GPS coordinates for user-defined flight paths. The license plate recognition pipeline running on the host computer is tasked with detecting and recognizing license plates in the incoming video stream. Plates are detected using an Adaboost classifier trained using Local Binary Patterns [4]. Characters are segmented using a combination of Sauvola binarization [5] and Connected Components Analysis. The open-source Tesseract OCR engine [6] is used to identify the segmented characters.

This thesis is structured in the following manner. A survey of license plate recognition techniques is presented in Chapter II. Chapter III contains an introduction to the Parrot AR Drone 2.0 platform. Each component of the proposed license plate recognition pipeline is described in Chapter IV. The design of the drone-based parking lot monitoring system is

described in Chapter V. The experimental studies performed to test the proposed system are presented in Chapter VI. Chapter VII contains the conclusions drawn from this work, along with ideas for future development.

# Chapter 2

# Literature Review

License plate recognition is defined as the process of locating and identifying the license plate of a vehicle in an image or video input. Due to immense variation in plate characteristics, background and operational environment, license plate recognition is considered to be a complex task. Recognizing a license plate involves three main subtasks: license plate localization, plate character segmentation and plate identification through optical character recognition of segmented characters.

The first stage involves locating potential license plates in images captured by a camera. Preprocessing is performed on the captured images in order to remove noise, blurring and lighting variations for improved performance. Classical image processing techniques such as edge gradient analysis, morphological operations and connected component analysis (CCA) are commonly used for plate detection. Due to the recent popularity of machine learning, classification algorithms such as Adaboost [7] and Support Vector Machines have been successfully adapted to solve the license plate location problem.

The second stage is license plate character segmentation, where the goal is to isolate the character regions from the rest of the detected plate. Methods commonly used for character segmentation include horizontal/vertical pixel projection, connected components analysis, Maximally Stable Extremal regions, component-based segmentation as well as a large va-

riety of locally adaptive binarization techniques such as Niblack and Sauvola. Machine learning techniques such as Adaboost and SVM, used in conjunction with feature descriptors such as Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG), have also been shown to achieve excellent character segmentation.

The final stage is plate identification through Optical Character Recognition (OCR). OCR is a highly mature topic in computer vision, with almost four decades of research and development. Many classical OCR engines are based on the artificial neural network (ANN). However, the development of kernel methods in the late $20^{th}$ century has led to the adoption of Support Vector Machine classifiers. OCR engines such as the ABBY FineReader [8] and Tesseract [9] are easily available along with several training sets for different fonts, languages and text styles.

The three subtasks (or stages) have been researched separately and as a group, resulting several license plate recognition system pipelines. This chapter serves as a survey of both classical and state-of-the-art techniques for each subtask.

## 2.1 License Plate Detection

License plate (LP) detection is the first stage in LP recognition systems. Preprocessing is performed to correct for noise, blurring and other disturbances before the extraction of potential LP regions in the image of the scene. LP detection methods can be broadly classified into image processing methods and learning-based methods.

Image processing methods are the classical approach to license plate detection. These methods are reasonably fast and require no prior training. However, they perform poorly on images with complex backgrounds and are prone to noise. Martin used edge statistics, gradient analysis and the top-hat morphological operator to locate potential LP regions [10]. In [11], [12], and [13], the Sobel edge operator is used to get the vertical edge statistics which are then analyzed to determine LP candidates. Guo et al. used high gradient averaging on

the Sobel Edge map to filter out most non LP regions. Final LP region(s) were obtained after connected components analysis and heuristic filtering. Hough line transform was used to correct perspective distortion [14].

Machine learning algorithms such as Adaboost and Support vector machines (SVM) are increasingly being used for plate detection as seen in surveys [15] and [16]. Statistical classifiers have very high detection rates, low false positives and small detection time. However, they require prior training with a decent dataset to achieve exemplar performance.

Adaboost based plate detection methods are based on the Viola-Jones object detection framework [17], which originally used Haar-like features and integral images for rapid face detection. Chen et al. [18] first adopted the Viola-Jones framework for scene text detection. Seventy-nine features, consisting of mean intensity, standard deviation, the x and y derivatives, histogram and edge linking features, were evaluated by a 4-layer cascade classifier. This classifier was trained using both individual and pairs of features as weak classifiers for the Adaboost meta-algorithm. Although not specifically trained for LP detection, the classifier was able to detect license plates due to the presence of text regions on the plate. Dlagnekov [19] performed Principal Components analysis on 1520 license plate images and determined that only 30 features were required to capture 90% of the textural and structural energy. $x$, $y$ derivative means and $x$ , $y$ derivative variances were used as the weak classifiers for the Adaboost classifier training to form class conditional densities for each feature. Although a high detection rate of 95% was achieved, character segmentation and OCR were not possible since a small window of 45x15 was used for the detector.

Limberger et al. [20] also followed a similar approach, choosing to use standard Haar-like features and edge orientation histograms for training the Adaboost classifier. Furthermore, a Kalman tracker was implemented to overcome low resolution images by combining information of the tracked plate from successive frames. Zheng [21] used Gentle Adaboost to train a two-classifier system which used Haar-like features and Histogram of Oriented Gradients[22] features.

Local Binary Patterns (LBP), proposed by Ojala et al. [4], are a class of highly efficient local textural descriptors which have been shown to match the performance of Haar-like features for Adaboost-based face detection, with the additional incentive of lower training times. Meijer incorporated local binary patterns (LBP) in his license plate recognition system as character descriptors for the character recognition part of the pipeline, opting to use a Histogram of Oriented Gradient features to train an SVM classifier for plate detection. In [23], Nguyen et al. developed a plate detection strategy most similar to the one proposed in this thesis. Haar features were used in conjunction with LBP features to train a classifier using Gentle Adaboost. This method claimed to merge the discriminative power of LBP with the intuition of Haar features to improve overall detection performance. However, this study was conducted using Vietnamese plates, which have black text on a white background. For plate detection, we propose to use only LBP features to get an optimal classifier that accommodates the variety in US license plates.

Support Vector Machines, developed by Vapnik and Chervonenkis [24], are a class of kernel-based supervised classification algorithms used to find a hyper plane that linearly separates a finite dimensional set of training points in a higher dimensional space. SVM use only a subset of the training points for the decision function (support vectors) and are generally quicker to train and evaluate as compared to Adaboost techniques. Therefore, SVM classifiers have been adopted by many researchers for the task of license plate detection [25]. A linear SVM classifier, trained using HOG features, was used by Li for plate detection [26]. However, the algorithm was computationally intensive due to the usage of HOG features. Another algorithm attempted to improve detection speed by using HOG-based Bag-of-Features to train a SVM classifier for LP detection [27]. Kim et al. decomposed an image into 16x16 grid cells and computed wavelet moments within each block before applying an SVM to classify text/non-text regions [28].

Maximally stable extremal regions (MSER) have also been adopted in some techniques to localize the license plate. MSER are a set of regions that are continuous and invariant to

affine transformations and monotonic changes in intensity [29]. One approach [20] modified the MSER framework to not only detect and track license plates, but to also perform character segmentation. Robust real-time tracking was achieved without any learning scheme. Kim et al. [30] used MSER to extract the local extremal regions, which were then filtered using boundary measures defined by LP text characteristics. Potential character regions were then classified using a Support Vector Machine. In [31], the MSER framework was used to find an initial set of regions which were then classified as plate/non plate by an SVM trained with Scale Invariant Feature Transform (SIFT) keypoint features. In order to overcome the inherent computational cost of SIFT, a class of nonlinear SVMs called Core Vector Machine was employed to successfully train and detect license plates with linear complexity.

Category specific extremal regions (CSER) were designed in [32] to detect plates by extracting local features from extremal regions, and then classifying potential LP character-like regions using a feed-forward neural network. This method achieved robustness against variations in illumination and viewpoint. In [33], prior knowledge of the plate layout was utilized to develop a component based LP detection method using conditional random field (CRF) model. Individual character-like regions were found using MSER and a CRF was constructed. LP bounding boxes were then estimated using belief propagation interference on the CRF. This method was found to capture both the spatial and textural relationship between characters.

Some other approaches have also yielded satisfactory performance for LP detection. In [34], the Stroke Width Transform [35] was modified to directly segment LP characters. Another approach [36] involved discovering the Principal Visual Word (PVW) to find LP using character geometric constraints and SIFT features. LP regions were detected by matching the detected PVWs with a PVW library trained in advance. Hsieh et al. [37] used the 1D wavelet transform to get the horizontal energy band. Horizontal projection and heuristic filtering are then applied to get a crude estimate of the plate location. Vertical projection and Black top-hat morphological operation are used for accurate plate localization.

## 2.2 License Plate Character Segmentation

Once a license plate has been isolated from the rest of the image, the next goal is to extract characters from the plate. Therefore, character segmentation is the next stage in the license plate recognition pipeline. It is be noted that Optical Character Recognition (OCR) is heavily reliant on the results of the character segmentation stage.

Binarization methods were originally developed to segment characters in scanned document images, which mostly have dark text on white backgrounds. However, character segmentation in license plates is a much harder problem due to the presence of complex backgrounds. Global thresholding methods such as Otsu's cannot perform optimal binarization for character segmentation, therefore research has mostly focused on using locally adaptive binarization methods such as Niblack and Sauvola [38]. However, due to recent interest in scene text understanding, several novel methods such as maximally stable extremal regions [39] and the Stroke Width transform [35] have been developed specifically to extract characters from scene images.

The classical solution to the character segmentation problem is the use of horizontal and vertical histogram profiles. In the work of Kim et al. [40], locally adaptive binarization was applied to the license plate image, followed by vertical pixel projection of the resulting binary image to extract characters. In order to tackle adverse effects at the binarization stage, prior knowledge of the plate was used by He et al. [41] to correct for plate inclination and illumination variation as well as for splitting and merging characters and edge enhancement. In the work of Yang et al. [42], the Laplacian transformation was applied to detect character edges and merged regions were binarized.

Morphological approaches to character segmentation have also been explored. In [43], morphological thickening and thinning were applied after Otsu's binarization to successfully segment overlapping character regions. Ezaki et al.[44] used the top-hat morphological operation to segment regions distinct from the background. Sobel edge operation is applied

separately to the red, green and blue color channels and the maximum Sobel edge values are combined. This image is then binarized using Otsu's method to yield character connected components.

Kim et al. [45] established a contour tracking algorithm which used boundary information of license plate characters to perform character segmentation. In [46], a novel image segmentation technique called the Sliding Concentric Window was used to determine local irregularity information which was then combined with Sauvola's binarization method. Kang [47] modelled an energy function consisting of the spatial and threshold characteristics which was minimized using dynamic programming to give rapid and accurate CS results. PhotoOCR [48], a system developed at Google, used a binary logistic classifier trained with a combination of HOG features and Weighted Direction Code Histogram features to segment characters from scene images with low resolution and complex background. A new HOG-based texture descriptor called T-HOG was proposed by Minetto et al. for classifying text/non text region using an SVM [49]. T-HOG was designed to characterize single-line texts in outdoor scenes by partitioning the image into overlapping horizontal cells with gradual boundaries.

Given the complex nature of license plates, traditional binarization methods fail to produce optimal results. LP character segmentation needs to be categorized as a scene text segmentation problem. Twelve different binarization schemes, including Niblack [50] and Sauvola [5], were evaluated by Milyaev et al. [51] based on OCR performance for scene text. A new binarization strategy was also proposed. A local binarization scheme was used to produce seed pixels which are labeled as strong/weak based on the magnitude of the Laplacian of the image. An energy function is constructed, consisting of a local energy term defined by the local binarization result and the Laplacian labeling process, and a smoothing term defined by neighborhood pixel similarity. The energy function was minimized globally using graph-cut interference to get the optimal threshold value for binarization. n [52], the scene text binarization problem was modeled as a Markov Random Field and a Gaussian Mix-

ture Model (GMM) was used to represent the background and the foreground. Since the GMM requires seeds, pixels are initially labeled using the Canny edge map. The foreground and the background models are learned using iterative graph-cut optimization, and a clean binarization result is obtained at convergence.

A nonlinear Niblack binarization method developed in [53] used two ordered statistics filters with the standard Niblack method to efficiently segment natural scene text. The resulting connected components were classified as text/non text using an Adaboost classifier trained using geometric and edge contrast features, in addition to shape regularity, stroke statistics and spatial coherence. Gatos et al. [54] used Sauvola's thresholding method to construct rough estimates of the foreground and background surface. A distance metric was used to compare the original image with the background surface to get the final threshold value. A team at Xerox designed a new binarization algorithm called Background Surface Thresholding [55] to improve OCR results for camera images. The foreground is iteratively thresholded using a function of mean neighborhood pixel intensities and background noise. This process removes foreground pixels and the empty areas are filled in using linear interpolation resulting in a background surface image. Finally, the original image is binarized using a pixel-wise threshold based on the background surface image.

Matas et al [32] proposed using a local optimal threshold value for each pixel to overcome problems due to non-uniform illumination. However, in [39] and [56], they postulated that separating the character segmentation from character recognition causes in loss of information. A new method to improve OCR performance was developed using MSER and two SVM classifiers. First, plate regions were classified using an SVM with a radial basis function kernel. Then, text lines were formed using an SVM with a polynomial kernel. After applying geometric normalization to correct for perspective distortion, an optimal image is fed into the OCR engine. [57] also used MSER but improved the segmentation performance by implementing an Extremal Region tree which captures the evolution of every connected component or extremal region across multiple thresholds. Irrelevant sub-

paths were filtered out by partitioning and pruning the tree. After some heuristic filtering, the remaining candidates are represented using local binary patterns (LBP) and Adaboost classifier is applied to segment character-like regions.

Some approaches attempted to utilize text color information for character segmentation. In [58], the Stroke Feature Transform was developed to incorporate color cues of text pixels along with the stroke features thereby improving inter-component separation and intra-component correction. Fraz et al. [59] recently proposed a new method of using color to enhance the text information. Illumination and reflection effects are removed using color enhancement. The color image is then quantized into N levels yielding N binary maps. The connected components obtained at each quantization level are filtered. HOG features are extracted and an SVM is used to classify text regions.

Kim et al. [60] developed a mobile application system which considered text location clues provided by the user. Using a mobile camera, a target region or window containing part of the text region is selected by the user. Colors in that region are clustered to give an estimate of the foreground color value, which is then used to perform binarization in the window. The text region is then iteratively expanded by searching the neighborhood for similar text color and updating the text color after every iteration. [61] introduced another mobile application for scene text recognition system using a new method called boundary clustering. Text and background surfaces are modeled and edge pixels are clustered based on color pairs and spatial position. Characters are extracted by assigning colors to the boundary layers after which stroke-based segmentation is applied.

## 2.3 License Plate Character Recognition

The most critical step in identifying the license plate is optical character recognition (OCR). OCR is widely considered a solved problem, having seen active research and development since the 1980s. Most OCR frameworks are built on the assumption that the characters or text lines originate from scanned documents. However, scene text exhibits

wide variation in orientation, font size, style and color. Therefore, OCR for scene text understanding is not yet a solved problem, with several new strategies published every year, particularly at the ICDAR Robust Reading competitions [62].

Early methods for OCR, as enumerated by Ning Li [63], used pixel-wise comparison to find the best matching character in a stored library. In one approach, the image was decomposed into 8x8 grids and those with strokes passing through them were identified as cellular features. A input image, decomposed to these cellular features, was then compared to a template library to find the best match. In another method [64], the stroke distributions were computed in the vertical and horizontal direction using projection and used for matching. In a further improvement, the image was decomposed to an m x n grid or mesh and the ratio of the stroke areas and the grid area was used as a feature for character classification. Such methods were sensitive to noise and character variety. Faster matching was achieved in [65], where a weighted feature-based hierarchical template evaluation technique was developed for license plate character recognition in order to effectively classify poorly segmented characters. [66] also presented a fast template matching method for license plate character recognition, where each character was scanned along a central axis to construct a feature vector containing the number of transitions between background and character.

In order to improve OCR speed by using only relevant image data, subsequent research focused on reducing the dimensionality of the input. Statistical classifiers such as K-Nearest Neighbors (K-NN), Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are used to evaluate the reduced input or feature vectors to effectively match characters. Dimensionality reduction and feature extraction were initially performed using transformations such as Fourier [67], Gabor and Karhunen-Loeve. [68] decomposed the character image using the Gabor wavelet transform and a 1-NN classifier was used for character recognition. Newell and Griffin [69] formulated features called oriented Basic Image Features (oBIFs) based on local orientation and symmetry and used Nearest neighbors for the character classification process. In [70], Newell et al. combined pairs of HOG features at different scales

to achieve robust character recognition using Nearest Neighbor classifiers.

Structural features such as contours and skeletons [63] have also been to improve the speed of OCR. A distance metric was used in [64] to fit the character contour to a given number of classes. [71] followed a graph matching approach where characters were represented using three types of features: curve point, branch point and end point. A similar approach was followed in [72] where graph grammar rules were used for character recognition in license plates.

The artificial neural network (ANN) was adopted in [73] to classify 64 x 1 feature vectors formed by decomposing the character image. Diep et al. [74] created a complex albiet more accurate neural network which classifies feature vectors of size 2500 x 1. Gonzanga et al. [75] used 17 features based on line slope, curvature, space interconnection, line interconnection in addition to other geometrical and topological features for representing characters in ANN training and classification.

Ramirez et al. [76] developed a structurally adaptive neural network which used Run Length Encoding to represent characters as feature vectors for ANN classification. In [77], Principal Components analysis was used for dimensionality reduction and the neural network classifier was trained using a radial basis function for the hidden nodes. [67] used an ANN to classify characters represented by Fourier descriptors. [78] and [79] improved recognition by using parallel neural networks to preserve the spatial compounding information. In [78], LP character features are extracted by skeletonization and normalized to a size of 8 x 16 pixels before classification using a parallel neural network. Canny edge detection operator and the blob coloring method were applied in [80] to separate LP characters and a feed-forward back-propagated neural network was used for character classification. In the work of Draghici et al. [81], two OCR engines were developed. The first was an ANN-based OCR engine which was used to classify feature vectors created by calculating the average intensity value in an 8 x 16 block. A second engine called the constraint-based decomposition was developed to perform classification and constructively improve OCR performance by updating the

network architecture. In [82], Histogram of Oriented Gradients (HOG) features are used to represent characters. Classification is done by a special category of ANN called Extreme Learning Machines in which the weights of the hidden layers are never updated.

Support Vector Machines have also been employed successfully for character recognition. SVMs are essentially binary classifiers while OCR is a multi-class problem. However, two strategies exist to adapt SVMs for character recognition: One-versus-All SVMs and One-versus-One SVMs. In the One-versus-All approach, the multi-class problem is reduced to several binary classification problems by training multiple SVM classifiers. For each classifier a positive label is assigned to the training examples of one class and negative label to the examples belonging to all the other classes. A winner-takes-all strategy is applied where the classifier with the highest output function is used to assign the output class. In the One-versus-One approach, each classifier assigns a vote to one of two classes and the class with the highest vote gets assigned to the output.

A comprehensive study of both SVM strategies is carried out by Thome et al. [83] based on their previous works for character recognition in Brazilian license plates [84][85]. The One-vs-One approach was found to have better classification accuracy (99.7%) as compared to the One-vs-All approach (96.7%). Liu et al. ( [86], [87]) found that SVM trained using chain code features and gradient features using the profile structure feature as a complementary feature, had accuracy rates over 99%. In [87], SVMs trained using Radial Basis Functions (RBF) or polynomial kernels were observed to have high levels of accuracy when used with proposed normalized cooperative feature extraction methods. [88] and [89] used multi-class SVM for character recognition in Indian license plates. Waghmare et al. [88] developed a One-vs-All SVM classifier system trained to recognize 36 classes (10 numbers and 26 letters). In constrast, Parasuraman and Subin [89] used One-vs-One SVM classifiers to perform OCR. Ahmad et al. [90] developed a hybrid OCR system which used RBF kernel based multi-class SVM and a Hidden Markov Model system for character and word level classification. In [91], Gabor features were used to construct a binary tree-like structure

using the SVM framework.

Xue [92] trained a One-vs-All SVM using Histogram of Oriented Gradients (HOG) features to classify hand-written as well as synthetic characters. C. Yi et al. [93] developed a new descriptor called Global HOG and used RBF-kernel SVM classifiers. When compared to local HOG descriptors, Global HOG demonstrated better modeling of the character structure, thereby improving OCR performance for scene text. Xin Li [26] and Bi Li et al. [94] investigated the use of HOG features and One-vs-All SVM classifier for recognition of USA and Chinese license plates respectively. G. Ning [95] extracted rotational HOG features separately from the RGB channels and trained a One-vs-All SVM for classification.

Local Binary patterns (LBP) have also found application in character recognition. Liu et al. [96] proposed a simplified LBP descriptor to produce optimal code patterns for Chinese LP characters. OCR is performed by matching the feature vector histograms using the Mahalanobis distance. Jayke Meijer [97] used the original LBP to decompose the character image into a feature vector and a linear SVM was used for character classification. In [98], a new descriptor called the local line binary pattern was formulated to encode the horizontal, vertical and magnitude information separately.

Some methods have been exclusively developed for scene text understanding. De Campos et al. [99] experimented with six types of features (shape context, SIFT, geometric blur, patch descriptor, spin image and maximum response of filters) and three different types of classifiers (SVM, k-NN and Multiple Kernel Learning) for scene text OCR performance. They found that a Multiple Kernel Learner trained using only 15 images had significantly better performance compared to the ABBY commercial OCR engine. Neumann and Matas [39] developed a robust scene text recognition system using maximally stable extremal regions (MSER). After normalizing the detected MSER regions to a fixed size, bitmaps are created by inserting boundary pixels based on orientation. An SVM classifier trained with a radial basis function is used to recognize characters.

Recent works have focused on using unsupervised and deep learning methods for character classification. Coates et al. [100] used a linear SVM for classifying features obtained from spatial pooling. Spatial pooling is an unsupervised feature extraction algorithm that combines the responses of a feature at multiple locations into one feature, thereby reducing the dimensionality of input character images. A related work by Wang et al. [101] used spatial pooling to select features which were fed into Convolutional Neural Networks (a class of massively networked ANNs) to achieve very high OCR performance. Character classification in the PhotoOCR engine [48] is performed by a deep neural network using HOG features.

To summarize, this chapter has explored almost two decades of research in the areas of license plate detection, character segmentation and text recognition. Machine learning methods such as Adaboost and SVM have found immense popularity for object detection. Adaboost in particular has been used to achieve excellent results with very low detection time. Feature descriptors such as Histogram of Oriented gradients (HOG) and Local binary patterns (LBP) have been used successfully for a variety of tasks including plate detection. License plate character segmentation is most commonly performed using locally adaptive thresholding methods such as Niblack and Sauvola. Finally, the wide availability of open-source Optical Character Recognition engines such as Tesseract emphasizes the maturity of character recognition techniques. Although not designed for scene text recognition, these OCR engines can perform reasonably well on optimally segmented characters. Several recent approaches have been aimed at scene text detection and recognition.

It has to be mentioned that almost all of the previous work on character segmentation and character recognition have exploited images with high discernibility between characters in addition to using high resolution images obtained from stationary cameras. It was observed that many license plate character segmentation and character recognition systems were developed for European or Asian plates which have dark letters on a white background, making the problem considerably easier. This work differs from previous approaches in

that we use low quality images from a moving camera, which introduces an additional level of complexity to plate detection and recognition. License plates being part of scene images, this work considers license plate recognition as a scene text understanding problem.

# Chapter 3

# The AR Drone

The AR Drone is a low cost quadrotor drone which was designed by engineers at Parrot, SYSNAV and MINES Paris Tech with the goal of creating a stabilized UAV platform at affordable costs for home entertainment applications such as video gaming and Augmented Reality (hence the prefix AR). The first AR Drone was released in 2010, followed by the AR Drone 2.0 in 2013. The AR Drone 1.0 was equipped with a 480p front camera, an ultrasound sensor and an inertial measurement unit (IMU) that measures pitch, roll, yaw and accelerations along all axes. The AR Drone 2.0, which is the platform used in this work, incorporated better features including a 720p front camera, a pressure sensor and better flight control due to the addition of a 3-axis magnetometer. A GPS/USB flash unit was also made available as an add-on feature to enable GPS-based navigation capabilities in addition to providing 4GB to record inflight video and navigation data.

The Parrot AR Drone can be remotely controlled by providing flight commands through Wi-Fi using a graphical user interface running on iOS or Android hand-held devices. Furthermore, an open-source AR Drone Software development kit (SDK) is provided for advanced users. The SDK can be used on Linux, Windows or Mac PCs to access drone hardware (camera and sensor data) and configure flight parameters (maximum tilt angles, altitude etc) thereby allowing implementation of custom flight control programs.

## 3.1 Basic Concepts in Quadrotor drones

The mechanical structure of a quadrotor drone consists of four rotors attached to the body frame. Opposite rotors are grouped into 2 pairs: $P_1(R_1,\ R_3)$ and $P_2(R_2,\ R_4)$. Both rotors in a pair turn in the same direction. However, the two rotor pairs P1 and P2 rotate in the opposite direction with respect to each other. For instance, P1 rotates counter clockwise while P2 rotates clockwise. Each rotor produces three types of dynamical forces:

**Thrust,** $T_i$ : A force that is generated by accelerating the quadrotor system in one direction, resulting in equal force on the system in the opposite direction.

**Torque,** $\tau_{\cdot}$ : A force that causes the quadrotor to rotate about its central axis,

**Drag,** $F_{drag}$ : A force that resists the quadrotor motion through the air, due to air friction.

The quadrotor moves as a result of differential torque and thrust. Pitch ($\Theta$), roll ($\Phi$) and yaw($\Psi$) are used to represent the angles of rotation in three axes about the vehicles center of mass. In order to stay airborne, the motors should generate sufficient vertical thrust to overcome the gravitational force in the downward direction. A vertical movement can be achieved by varying the motor thrusts ($T_1$,$T_2$,$T_3$, $T_4$) by the same magnitude. This changes the total vertical thrust without creating a differential torque on the body, thereby avoiding rotation along any of three axes. Difference in the angular velocity of each rotor pair creates an angular acceleration ($\dot{\Psi}$) about the yaw axis. Each rotor pair can be used to control the pitch and roll angles separately. Increasing thrust for one rotor while decreasing thrust for the other rotor in the pair introduces a differential torque about the pitch or roll axes, while maintaining stability in the yaw axis. Horizontal forward/backward motion can be initiated by creating a differential between the front and the rear rotors, resulting in a non-zero pitch angle $\Theta$. Similarly, a sideways movement can be obtained if the roll angle $\Phi$ is non-zero as a result of differential torque between the left and the right rotors.

$\Theta$ and $\Phi$ govern the magnitude of acceleration of the drone along the pitch and roll axes. The greater the tilt, the faster the drone will accelerate. Therefore, by placing an upper

bound on the tilt angles, an upper bound can be placed on the maximum acceleration of the drone. However, if this acceleration is insufficient to overcome air resistance, the drone will not move in the desired direction.

## 3.2 Previous work using the AR Drone

Due to low cost and high modularity, the AR Drone has become an attractive choice for researchers in academia and industry [102], resulting in its adoption for a variety of applications. Bills et al. [103] used perspective cues to navigate structured indoor environments. Another method [104] used SURF features and template matching for indoor navigation of the AR Drone using its front camera. Vissier et al [105] developed advanced navigation capability for the AR Drone based on simulation. In [106], probabilistic models for sensing and motion were used to learn elevation maps, optimal paths and optical flow based obstacle avoidance. [107] used an extended Kalman filter (EKF) to improve localization accuracy to within 10cm by fusing information from GPS, Inertial Measurement Unit (IMU) and Ultra-wide band (UWB) tracking.

Cremers et al. published several works on camera based navigation using the Parrot AR Drone. In their first work [108], a monocular SLAM-based navigation system was developed for GPS-denied indoor environments. Data fusion and state estimation were performed by an extended Kalman filter and a PID flight controller was used. Following a similar approach, state pose estimation was used to achieve accurate figure flying in an indoor environment [109]. In a more recent work [110], scale estimation was attempted using the front camera to assist in visual SLAM for navigation.

## 3.3 Hardware

The body of the AR Drone 2.0 [111] is constructed using carbon fiber and high resistance plastic. The rotors are made with fiber-charged nylon plastic parts. Two light weight hulls made of Expanded Polypropylene (EPP) foam are also provided with the drone

to protect against mechanical shocks during indoor and outdoor flights. The total weight of the drone is between 380-420g, depending on the hull being used.

The AR Drone 2.0 is equipped with a 1GHz 32-bit ARM Cortex A8 processor and a dedicated video processor (800MHz DSP TMS320DMC64x). The processors run Linux 2.6.32 as the default operating system. 1 Gbit 200 MHz DDR2 random access memory (RAM) and a 802.11g wireless card are also integrated with the processor to provide computational power and Wi-Fi connectivity. Custom software can be flashed to the board through a universal serial bus (USB) connection, which can also be used to attach add-on units like the GPS.

The four rotors are powered using 14.5W brushless DC motors, which have a range of 10,350- 41,400 RPM. While hovering, each motor runs at 28,000 RPM which translates into 3,300 RPM on the rotor by using the 1/8.75 Nylatron reductor gears. A custom Electronic Speed Controller (ESC), which consists of a low power 8-bit AVR microcontroller and a 10-bit ADC, is used to control each motor. A protection system continually monitors all the motors. It prevents damage by cutting off power to all the motors if any rotor encounters an obstacle.

The standard power source for the AR Drone 2.0 quadrotor system is a 1000 mAh Lithium polymer (LiPo) battery a flight time of approximately 10 minutes. However, 2000 mAh LiPo batteries were used in this work to allow flight times up to 20 minutes.

## 3.4 Sensors

A variety of sensors are incorporated with AR Drone 2.0 in order to allow automatic stabilization and to provide video data for the user. Flight dynamics are monitored by the various sensors and sensory data is analyzed by internal stabilization algorithms to allow robust control and stable flight. Take-off, hovering, trimming and landing are fully automatic using sensory data [112].

The AR Drone features a 6 degree-of-freedom inertial measurement unit (IMU) which provides the pitch, roll and yaw measurements necessary for flight stabilization and assisted tilting control. The IMU contains a 3-axis accelerometer, a 2-axis roll/pitch gyroscope and a single axis yaw gyroscope. One or more accelerometers are used to detect the current rate of acceleration of the inertial frame with respect to the world frame. The gyros are used to measure angular velocity in degrees per second. The angular velocity signal is integrated with respect to time in order to estimate the absolute angle. However, sensor noise leads to orientation drift. The AR Drone 2.0 uses a 3 axis magnetometer to correct for this orientation drift. The IMU allows motion estimation and stabilization [112].

An ultrasound telemeter is used to measure the altitude up to 6m [113] after which the altitude is measured by a pressure sensor. The altitude measurements are also used for stabilization.

Finally, two vision sensors are present on the AR Drone: a QVGA 240p vertical camera and a 720p 30frames per second (FPS) front camera. The vertical camera has a field-of-view of 64° and is used to measure the ground speed for automatic hovering and trimming. The front camera consists of a CMOS sensor with 92° lens. Live video feed from either camera can be relayed over Wi-Fi or recorded on a USB flash drive using JPEG/H264 encoding scheme. For this work, the front camera was tilted by 16° along the pitch axis so that the drone can obtain a direct view of vehicle license plates while maintaining a safe flying altitude.

## 3.5 Software

The open-source AR Drone Software Development Kit (AR Drone SDK 2.0) allows the development of custom software for the drone using C. The AR Drone Library is part of the SDK and provides high-level application programming interface (API) for developing custom applications. It includes the details about the video processing pipelines,

codecs, input device management, communication and navigation structures. It also allows implementation of custom controller threads for flight control.
Communication between the AR Drone and the client is through an ad-hoc Wi-Fi connection. There are four main services implemented in the SDK [112].

1. **ardrone_tool** or **AT commands**: Transmitting/receiving data at a frequency of 20Hz, this service is used to control the drone. This thread will be used by the flight control loop (see Section 3.7.3) to send input commands to the drone. The syntax of the control input string is shown below,

```
ardrone_tool_set_progressive_cmd (
    int32_t control_mode,
    float32_t roll_angle,
    float32_t pitch_angle,
    float32_t vertical_speed,
    float32_t yaw_speed,
    float32_t mag_psi,
    float32_t mag_psi_accuracy )
```

   The first argument is used to switch between hovering mode (0) and control mode (1). The next two arguments are the roll angle $\Phi$ and pitch angle $\Theta$ between (-1, 1) where 1 refers to the maximum preset value. The drone tilts left and forward for negative values of $\Phi$ and $\Theta$ respectively. Positive vertical speed increases drone altitude. Left/right rotation along the yaw axis is obtained by specifying negative and positive values respectively. The last two arguments are used to refine the compass heading using the magnetometer. The application programming interface also allows specification of drone flight environment (indoor/outdoor), protective hull type (indoor/outdoor shell) and other configuration data.

2. **Navdata**: This service receives various instrumentation data including sensor measurements at a frequency of 30Hz (200Hz in debug mode). This data is unpacked by

the SDK and made available to the user in the form of structures, the most important of which are listed below.

```
navdata_demo_t \\ Battery status, Velocities V_x, V_y, V_z
navdata_time_t          \\ Time
navdata_euler_angles_t  \\ Theta, Phi, Psi
navdata_altitude_t      \\ Altitude
navdata_magneto_t       \\ Magnetometer readings
navdata_gps_t           \\ GPS coordinates
```

3. **Video**: This service allows live streaming of video data from the drone using the Parrot video encapsulation and the MPEG4 codec. Video packets received from the drone are decoded by the SDK and direct user access is possible through the API. This video data will be processed by the host computer to perform license plate recognition. Using the **ardrone_tool** commands, frame rate and bitrate of the video stream can be adjusted between 15-30 frames per second and 300-1700 bits per second respectively.

4. **Control Port**: Critical tasks such as configuration data retrieval are communicated through this channel via TCP/IP for reliability.

## 3.6 Control

The internal flight controller of the AR Drone is based on the difference between the user input and the current values of the controllable parameters. Values in the user-defined input string (see Section 3.5) should be in the range (-1, 1) as a factor of preset maximum values [112]. The pitch ($\Theta$) and roll angles ($\Phi$) can be specified to control the linear and lateral velocities of the drone. Rotation about the yaw axis can be achieved by specifying yaw speed ($\dot{\Psi}$). The altitude is controlled by providing the desired vertical speed ($V_z$) in the input string.

In order to ensure stable and controlled motion, the drone does not allow the user to directly control the motors. This task is performed by an internal control system which regulates the motor speeds based on the **ardrone_tool_set_progressive_cmd** arguments. Different modes of the drone (flight, take off etc) are modeled as a finite state machine. There are two nested control loops: the Attitude Control Loop and the Angular Rate Control Loop [113].

**Attitude Control Loop:** In this loop, the difference between the estimated heading and the desired heading is used to compute an angular velocity for the motor which is tracked using a Proportional Integral (PI) controller. The Attitude Control Loop also handles altitude stabilization using information from the ultrasound sensor to maintain a fixed distance from the ground. However, the ultrasound sensor data may cause the drone to automatically increase its height due to perceived difference in altitude when the drone flies over objects. To overcome this undesirable behavior, the altitude stabilization is based on a filtered derivative of the ultrasound measurements [114]. The ultrasound sensor is also used during take-off to reach a fixed altitude quickly, and to decrease the vertical velocity $V_z$ when near the floor while landing.

**Angular Rate Control Loop:** This loop consists of simple proportional (P) controllers for motor speed control. If no user input is specified, a Hovering Control Loop is used to maintain constant altitude and zero speed using a PI controller.

# Chapter 4

# The License Plate Recognition Pipeline

This section introduces the license plate recognition pipeline. The algorithms involved in license plate detection, pre-processing, character segmentation and OCR are described in detail. OpenCV 3.0 was used to implement the computer vision algorithms because of its immense functionality and ease of use, in addition to being open-source.

## 4.1 License plate detection

This work adopts the Viola Jones object detection framework to rapidly locate license plates in video frames. However, instead of the standard Haar-like features, local binary pattern (LBP) descriptors are used to represent images. Local histograms are first extracted from license plate images using an LBP operator. Next, a spatially enhanced feature vector called the LBP histogram is obtained by concatenating the local histograms. These feature vectors are used to train a Gentle Adaboost classifier.

### 4.1.1 Local Binary Patterns

Local binary patterns (LBP) are a class of powerful, computationally simple textural descriptors which are used to capture local image structure by evaluating pixel neighborhood. The LBP histogram of a region can serve as an effective texture descriptor since the LBP operator is invariant to monotonic gray-scale transformations (Figure 4.3).

Local binary patterns are computed by taking the central pixel location as the reference point and its value ($I_c$) as a threshold for comparison with its neighbors. If a neighboring pixel has an intensity ($I_p$) higher than the threshold, it is assigned a value of 1, else it is assigned a value of zero. Following a binary code construction scheme as shown in Figure 4.1, each pixel in the input image can be represented by a binary number or the local binary pattern code. LBP codes are written as $LBP_{(P,R)}$ where P is the number of neighbors and R is the radius of the neighborhood within which the image structure is evaluated. The construction of an 8-neighbor local binary pattern code is defined in Equation 4.1,

$$LBP_{(8,1)} = \sum_{p=0}^{8} n(I_p - I_c)2^p \tag{4.1}$$

where $n$ is determined by the difference in intensities between the neighbor and the central pixel.

$$n = \begin{cases} 1 & \text{if } (I_p - I_c) \geq 0 \\ 0 & \text{if } (I_p - I_c) \text{ ¡ } 0 \end{cases}$$

The original local binary pattern fails to encode large textural features which are necessary for multi-scale description [115]. This was addressed by using a circular neighborhood operator of radius $R$, consisting of $P$ evenly spaced points. Parameters $P$ and $R$ regulate angular space quantization and spatial resolution respectively. Multi-resolution analysis can be performed by combining the responses of multiple operators with different ($P$, $R$) pairs. Neighboring pixel locations $(x_p, y_p)$ are estimated as follows:

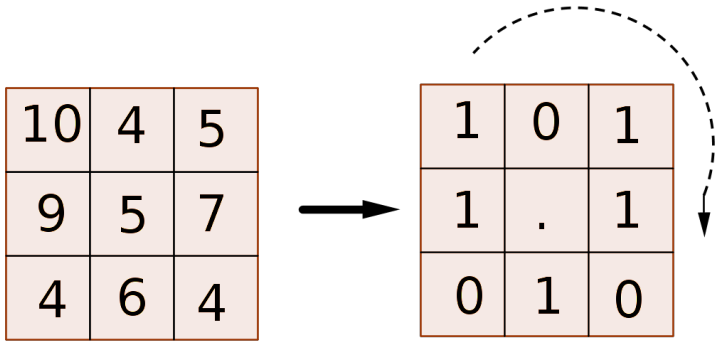


Figure 4.1: Local Binary pattern code formation


Figure 4.2: SC *(left)* and NC license plates *(center)*, both subjected to $LBP_{(8,1)}$ and SC plate subjected to the circular $LBP_{(16,2)}$

$$x_p = \quad x + Rcos(2\pi p/P) \tag{4.2}$$

$$y_p = \quad y - Rsin(2\pi p/P) \tag{4.3}$$

The LBP code for the circular LBP (also called the extended or E-LBP) is defined as follows:

$$LBP_{(P,R)} = \sum_{p=0}^{P} n(I(x_p, y_p) - I_c)2^p \tag{4.4}$$

In this license plate recognition pipeline, an advanced LBP derivative called the multi-scale

Figure 4.3: LBP invariance to monotonic changes in intensity

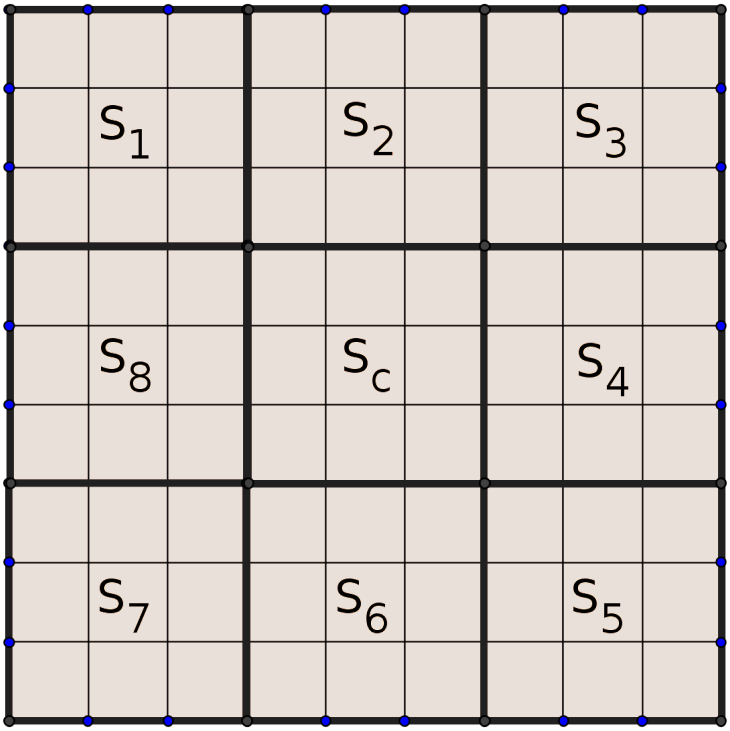


Figure 4.4: Multi-Block Local Binary Pattern for scale $s = 9$

block local binary pattern (MB-LBP) is used for encoding license plate images into feature vectors. MB-LBP efficiently represents the image by incorporating the macrostructure of the image with its microstructure[116]. The MB-LBP operator divides the image into $s$ x $s$ sub-regions or blocks where $s$ is the scale (Figure 4.4). Within these blocks, the average

sum of image intensities ($S_1$-$S_8$) are computed rapidly using integral images. The average sums of the $P$ neighbors block are compared to the central block sum ($S_c$) to form the MB-LBP code. MB-LBP thus effectively encodes the local information in each block while eliminating noise due to the large window size. The local histograms extracted from each block are concatenated to form the final LBP histogram which is used as input to the classifier.

### 4.1.2 Adaboost

Boosting is a machine learning approach where the goal is to create a strong classifier by combining several weak(inaccurate) classifiers or learners. The only selection criterion for the weak classifiers that the accuracy of each classifier is slightly better than random guessing. Therefore, for a 2-class problem like plate detection (plate/non-plate), each weak classifier must have an accuracy greater than 51%. Adaboost (Adaptive Boosting) is the most popular boosting algorithm as it can achieve excellent classification performance when trained properly. Decision trees are the most common weak learners used for classifier training with Adaboost.

Given a labeled training set $\{(x_1, y_1)...(x_m, y_m)\}$, weak classifiers can be constructed by considering individual or combinations of input features. These weak classifiers ($h_i(x)$) are iteratively weighted ($w_t$) and combined to minimize training error ($E_t$) at every stage $t$, resulting in a potentially strong classifier $f_t(x)$:

$$f_t(x) = w_t h_i(x) \tag{4.5}$$

Different variants of Adaboost (Real Adaboost, Discrete Adaboost, Gentle Adaboost) use different training error metrics to choose $f_t(x)$. Gentle Adaboost uses least squares fitting to minimize the error function ($E_t$) as shown in Equation 4.6.

$$E_t = \sum_{n=i} w_{t,i}(y_i - f_t(x_i))^2 \tag{4.6}$$

After several stages of training, the final boosted classifier ($F(x)$) is formed as the sum of the strong classifiers,

$$F(x) = \sum_{t=1}^{T} f_t(x) \tag{4.7}$$

Viola and Jones [17] laid the foundation for a new object detection framework based on boosting. Firstly, image features are computed using integral images with linear complexity. Next, Adaboost is used as a feature selection process where each weak classifier is dependent only on one image feature. The resulting classifier retains only strong features since irrelevant features are quickly eliminated by Adaboost. Algorithm 1 describes the feature selection process using Gentle Adaboost. The resulting classifiers are combined into a degenerate decision tree-like structure, resulting in a cascade of classifiers. This improves detection speed and performance since large areas of the image are discarded by the initial classifiers using relatively simple features. Subsequent classifiers in the cascade evaluate computationally intensive features to produce the final classification result. The cascade classifier can be optimized by varying the number of features used, the number of classifiers in the cascade and the threshold criterion at each stage of classifier training.

### 4.1.3 Implementation

The plate detector was created using the Viola Jones framework with Gentle Adaboost for feature selection. LBP histograms are computed by applying the multi-scale local binary pattern (MB-LBP) operator to the integral image. These features are used to train a Gentle Adaboost cascade classifier with 14 layers. Decision stumps, which are one-level decision trees, were used as weak learners. Thirty-two features were learned by the classifier at the end of the training process. Few of the more interesting features are shown in Figure 4.5). It can be seen that several features are learned around the tree in SC plates, in addition to dark-to-light and light-to-dark transitions which characterize the presence of text in an image.

At the end of Adaboost training, an XML file containing the classifier information is gener-

**Algorithm 1** Gentle Adaboost learning algorithm used in the Viola Jones framework

1. Given a training set $\{(x_1, y_1)...(x_n, y_n)\}$ where $y_i = 1$ or 0 for positive and negative examples respectively.

2. Initialize weights $w_i = 1/2m$, $1/2l$ for positive and negative examples respectively, where m and l are the number of positive and negative samples.

**for** $t = 1$ to $T$ **do**

**for** each feature $j$ **do**

3. Train weak classifier $h_j$

4. Minimize classifier error $e_{j,t}$ using least squares fitting

$$e_{j,t} \Leftarrow \sum_j w_{j,t}(y_i - h_j(x_j))^2 \tag{4.8}$$

**end for**

5. Select classifier $h_{j,t}$ with the smallest error $e_{j,t}$ as strong classifier $f_t$ for stage $t$.

6. Update the weights

$$w_{j,t} \leftarrow w_{j,t}\beta_t^{1-e_j} \tag{4.9}$$

where $\beta_t = e_{j,t}/1 - e_{j,t}$ and $e_j = 1$ or 0 if classified correctly/incorrectly.

7. Normalize weights to create a probability distribution:

$$w_{j,t} \leftarrow w_{j,t}/\sum_{i=1}^{n} w_{j,t} \tag{4.10}$$

8. Final boosted classifier

$$F_t(x) \leftarrow F_{t-1} + f_t \tag{4.11}$$

**if** termination criteria reached **then**
$F_t(x) \leftarrow$ Final boosted classifier
**end if**

**end for**

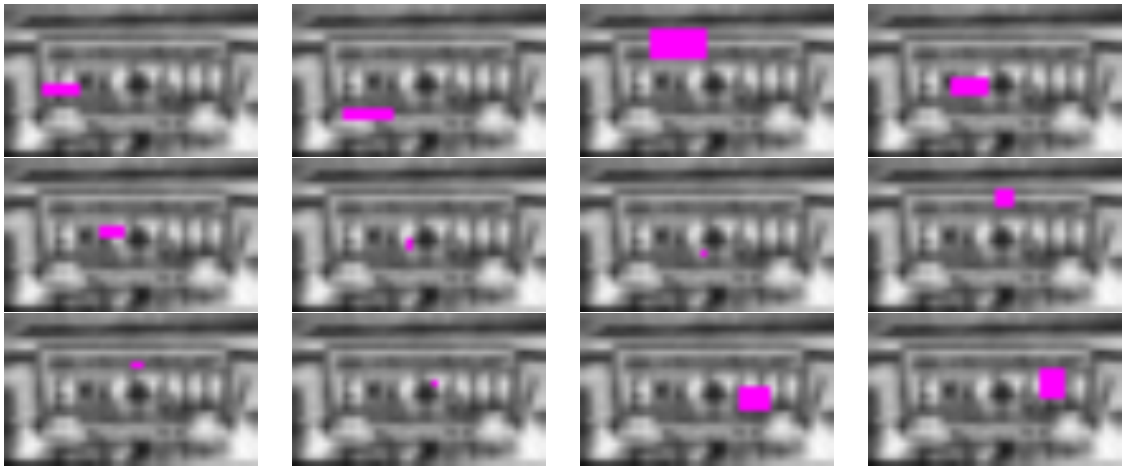

Figure 4.5: Features learned by the cascade classifier during training

ated. During operation, file is loaded by the license plate recognition pipeline to instantiate a plate detector object. Video frames are converted to grayscale before being evaluated by the plate detector.

## 4.2 License plate correction

Once a plate has been identified, it is subjected to various corrective operations in order to improve performance at the character segmentation stage. Detected plates are corrupted by significant noise, uneven illumination and perspective distortion. Pre-processing plays a vital role in improving image quality for better character segmentation and recognition results. This section describes the pre-processing methods implemented in the license plate recognition pipeline.

Figure 4.6 shows a template of the South Carolina license plate and an image captured by the drone. It can be seen that the image received from the drone is affected by the low quality camera sensor and low bitrate of the video stream. The presence of unfavorable illumination and motion blur further corrupt several frames, therefore only a small fraction of the detected plate images are available for further processing. Severely blurred plates are discarded by using the variance of Laplacian of the image as a measure of focus[117].


Figure 4.6: SC license plate template *(left)* and SC license plate captured by the drone camera *(right)*.


Figure 4.7: License plate captured by the drone *(left)* and license plate image after Gamma transformation *(right)*

### 4.2.1 Gamma correction

Gamma correction is a non-linear image transformation which is used for contrast enhancement. Contrast enhancement may be defined as the process of increasing the discernibility between light and dark regions in a grayscale intensity image. Given an input image pixel $I_{in}(x, y)$, the standard gamma correction operation can be defined as,

$$I_{out}(x, y) = 255 \left( \frac{I_{in}(x, y)}{255} \right)^{\gamma} \tag{4.12}$$

where $\gamma$ (as seen in Equation 4.13) adjusts the dark ( $I_{in}(x, y)$ <60 ) and bright ($I_{in}(x, y)$ >190) pixels more as compared to the median pixel intensities.

$$\gamma(x, y) = 2^{\left(\frac{I_{in}(x,y)-127}{128}\right)} \tag{4.13}$$

Gamma correction in this license plate recognition pipeline is performed using the locally adaptive method proposed in [118]. The image histogram is partitioned based on local

minima and the mean gray-level ($I_{mp}$) of each partition is used for gamma correction as seen in Equation 4.14,

$$\gamma(x,y) = 2^{\left[1-\frac{255-I_{in}(x,y)}{128\alpha+(1-\alpha)I_{mp}}\right]} \tag{4.14}$$

The factor $\alpha$ in Equation 4.14 is a function of the frequencies of the local histograms. Figure 4.7 shows the effect of applying histogram partitioning based gamma correction to the detected plate image.

### 4.2.2 Color channel processing

Given the geographic location of the experiments, South Carolina (SC) plates were the most frequently encountered class of license plates. SC license plate (Figure 4.6) contain a color gradient which complicates the character segmentation problem. During initial experimentation, it was found that the orangish hue in the lower regions of the plate had an adverse effect on character segmentation. Therefore, separate processing of the red, blue and green channels was considered. Satisfactory segmentation results were obtained for SC plates by using only the red channel while no adverse effect was observed in other plates (Figure 4.8). It can be observed that some noise gets added as a result of using only the red channel but the characters are not broken.

### 4.2.3 Perspective correction using the Hough Transform

Detected plates are subjected to perspective transformation since the drone camera experiences changes in viewing angle. Therefore, it is necessary to undo this perspective transformation in order to simplify character segmentation. In this system, rotation along the image plane is rectified by applying a corrective transformation using a scaled rotation matrix $R_s$ is defined as:

$$R_s = \begin{bmatrix} \alpha & \beta & (1-\alpha)x - \beta y \\ -\beta & \alpha & \beta x + (1-\alpha)y \end{bmatrix} \tag{4.15}$$


Figure 4.8: Result of binarization for Georgia (*left*) and SC plates: using the grayscale image (*left*) and the red channel only (*right*)

where $\alpha$ and $\beta$ are defined by rotation angle $\theta_r$ and scale $s$ as follows,

$$\alpha = s \cos \theta_r$$

$$\beta = s \sin \theta_r$$

The Hough transform is feature extraction technique commonly used as a shape detector, particularly for detecting lines in an image. It uses a voting scheme to effectively find imperfect instances of a shape in an image. Equation 4.16 defines a line in parametric form, where $\rho$ is the perpendicular distance to the origin of the line and $\theta$ is the angle between the perpendicular line and the horizontal axis.

$$\rho = x \cos \theta + y \sin \theta \tag{4.16}$$

Using $(\rho, \theta)$ as the parameter space, the Hough transform detects line points as local maxima in an accumulator space where votes are counted for each $(\rho, \theta)$ pair applied to all non-zero points in the edge map. This formulation of the Hough transform is computationally intensive since it evaluates a large number of points. The Progressive Probabilistic Hough transform increases computational efficiency considering a random subset of the edge points[119].

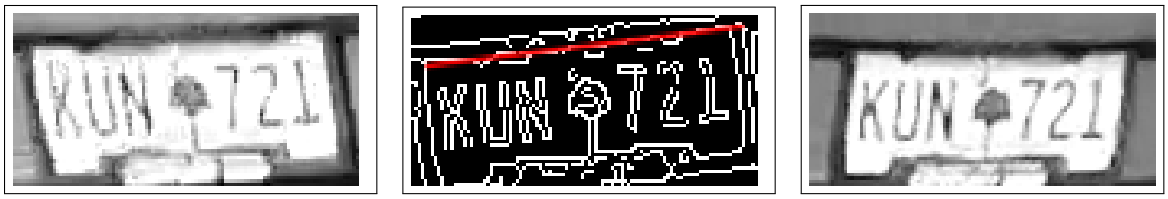


Figure 4.9: From left: Distorted license plate, longest line detected by the Hough Transform in the Canny edge map, corrected license plate image

In this work, lines extracted from the license plate image are used to determine the angle of tilt along the plane. First, the probabilistic Hough transform is applied to locate the lines in the Canny edge map of the plate image. The upper and lower boundaries of the license plate are usually the longest line candidates detected by the Hough transform (Figure 4.9). The angle ($\theta_r$) needed for the rotation matrix $R_s$ in Equation 4.15 can be computed using the end points of the longest line candidate. In OpenCV, the probabilistic Hough transform function directly returns the endpoints of the detected lines. After computing the tilt angle $\theta$, the image is subjected to perspective correction using the matrix $R_s$. Figure 4.9 shows a license plate image before and after perspective correction.

## 4.3 License plate character segmentation

The next stage in the LP detection pipeline is character segmentation. Character segmentation involves extracting characters from license plate images. This section describes the character segmentation stage of the license plate recognition pipeline. License plate characters are extracted using locally adaptive binarization and connected components filtering. After perspective correction, the plate image is scaled up to a size of 300 x 150 using bicubic interpolation in order to enhance inter-character separation.

### 4.3.1 Binarization

The goal of binarization is to find a threshold $t(x, y)$ such that

$$I_{out}(x,y) = \begin{cases} 255 & \text{if } (I_{in}(x,y)) \geq t(x,y) \\ 0 & \text{if } (I_{in}(x,y)) < t(x,y) \end{cases}$$

where $I_{in}$ and $I_{out}$ are the input and output image respectively. Figure 4.10 shows the performance of different thresholding methods on a test image. It can be seen that global thresholding methods such as Otsu's do not satisfactorily perform character segmentation. Therefore, locally adaptive binarization methods such as Niblack and Sauvola were considered.

Niblack [50] and Sauvola [5] methods use statistical information such as mean ($m(x,y)$) and standard deviation ($s(x,y)$) from a local neighborhood window $W_x$ x $W_y$ around the pixel $I_{in}(x,y)$. In this pipeline, we calculate these statistics using integral images, which drastically reduces computation time. Niblack threshold $t_n(x,y)$ can be computed as

$$t_n(x,y) = m(x,y) + k \cdot s(x,y) \tag{4.17}$$

where $k$ is a weighting constant usually in the range (0-0.5). Sauvola modified Equation 4.17 as follows:

$$t_s(x,y) = m(x,y)\left[1 + k(\frac{s(x,y)}{D} - 1)\right] \tag{4.18}$$

where $D$ is the dynamic range. In this work, $D$ is computed as the difference between the maximum and minimum intensity values in the neighborhood. For the task of character segmentation in this plate recognition pipeline, Sauvola binarization is used with window sizes $w_x$=13, $w_y$= 17 and weight $k$= -0.15. Figure 4.11 shows the effect of varying window sizes ($w$) and $k$ on binarization using Sauvola's method.

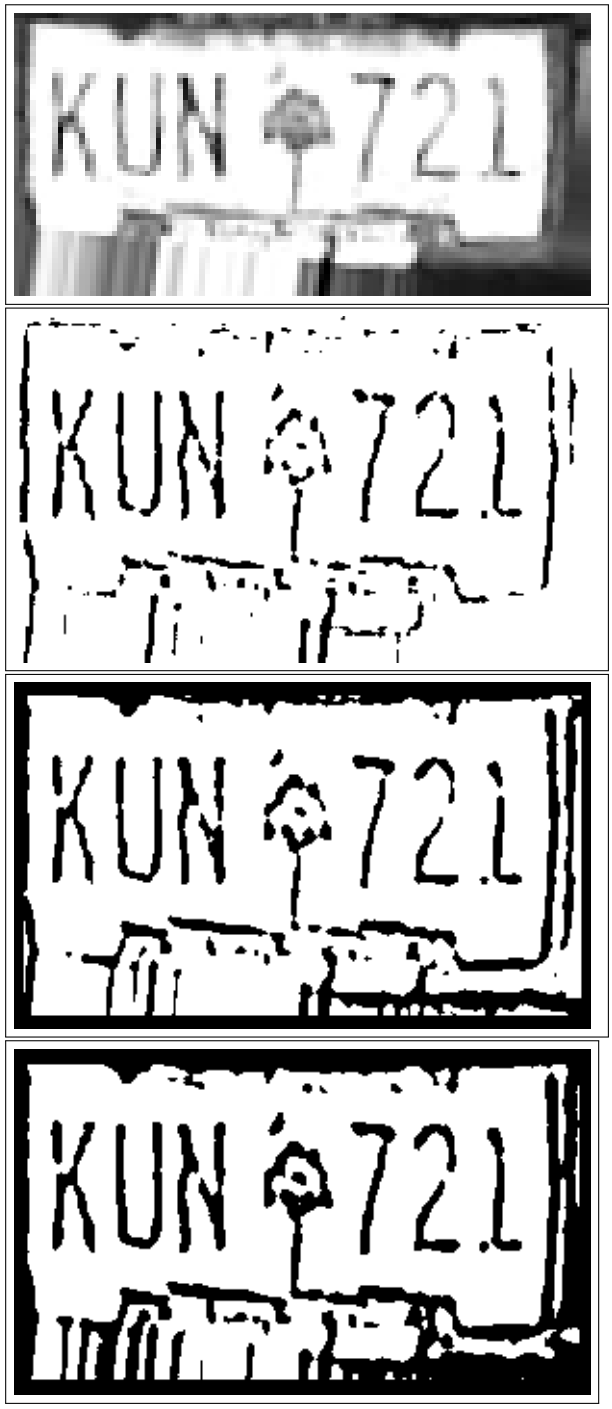


Figure 4.10: From top: sample license plate, Otsu's global thresholding, Niblack's method and Sauvola's method

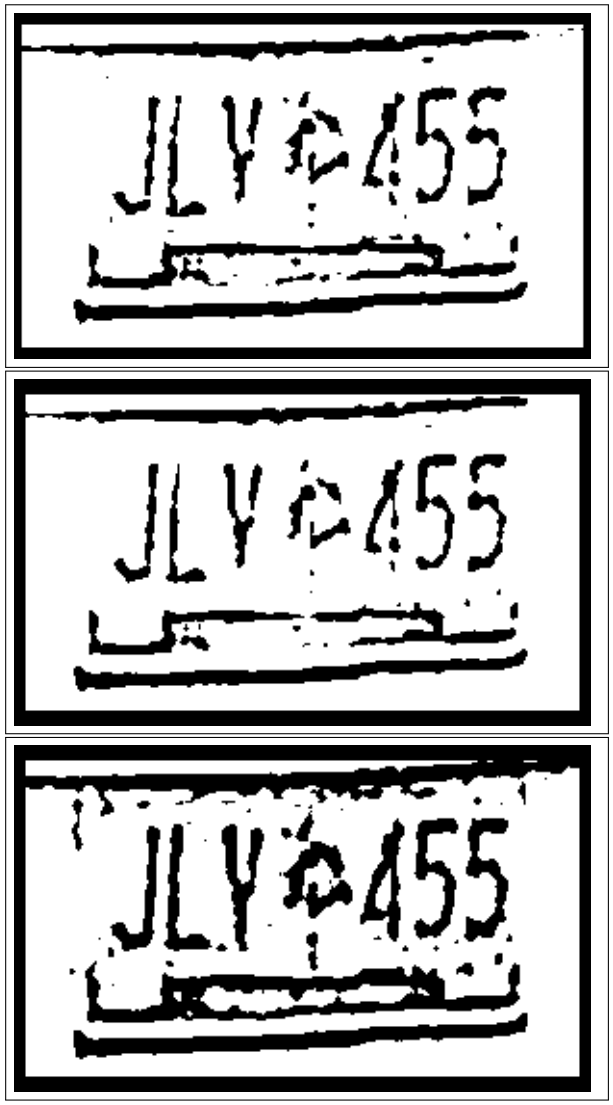

Figure 4.11: Sauvola binarization results: (from top) small window size ( $w_x = 7, w_y = 11$) with ($k = -0.15$), large window size( $w_x = 13, w_y = 17$) with large weight ($k = 0.10$), large window size ( $w_x = 9, w_y = 13$) with small weight ($k$ = -0.15)

### 4.3.2 Connected Components Analysis

Binarization results may contain several unwanted areas including plate edges and broken pieces of the background. Connected components analysis can be used to filter out non-character areas. True character regions can be extracted using heuristic filtering based on component area and location (Figure 4.12).

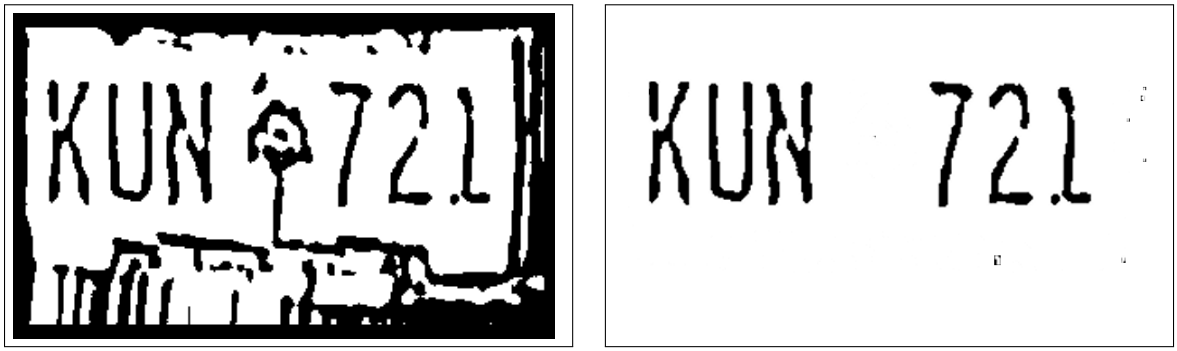


Figure 4.12: Binarization result before and after connected component analysis and filtering

## 4.4 Optical Character Recognition

The final stage in the license plate recognition pipeline is plate identification through optical character recognition (OCR). The output image of the connected component filtering stage consists of mostly character blobs and can directly used for character recognition

Tesseract [6] is an open-source OCR engine that was originally developed at HP. The current version (Tesseract 3.0.3) is maintained by Google. The Tesseract optical character recognition pipeline is as follows:

- Blobs are formed by analyzing the connected components in the input image.
- Blobs are then sorted based on horizontal position and assigned to unique text lines, without being affected by image skew
- Text pitch is evaluated in order to chop fixed pitch words into characters. A fuzzy approach is employed for separating words and characters with varying pitch.
- A fully-chop-then-associate approach is used to resolve joined or broken characters. Separation of joined characters is performed by chopping at concave vertices in the outline.
- 50 to 100 features (represented by position$(x, y)$ and angle) are extracted from the short and thick lines in the polynomial approximation of the characters. These 3-

dimensional features are matched with 4-dimensional (position, angle, length) prototype features .

- Character classification takes place over two passes. In the first pass, a static classifier shortlists potential character classes from the set of all possible classes. The second pass consists of an adaptive classifier trained using the output of the static classifier in order to recognize application-specific fonts.

Tesseract performs best when used for its original purpose: optical character recognition of scanned text. However, a number of recent works have adopted Tesseract for scene text recognition ([51],[120],[121],[122],[123]). The quality of binarization is of utmost importance as Tesseract is meant to operate on clear, definite characters encountered in high resolution scanned document images. Unfortunately, scene text seldom exhibits the same behavior as printed document text. Therefore, when Tesseract is used for scene text recognition, factors such as noise, blur and text variety adversely affect OCR performance.

OpenCV provides a text detection module with embedded support for the Tesseract OCR engine. The images containing segmented LP characters or text blocks are directly processed in the pipeline using OpenCV-Tesseract. Figure 4.14 shows license plate character recognition using OpenCV-Tesseract on a binary image obtained after character segmentation stage. Small, broken components which escape CC-based filtering stage affect the recognition since Tesseract is programmed to find punctuation marks (Figure 4.13). Therefore, Tesseract's vocabulary was manually restricted to recognize only capital letters and numerals.

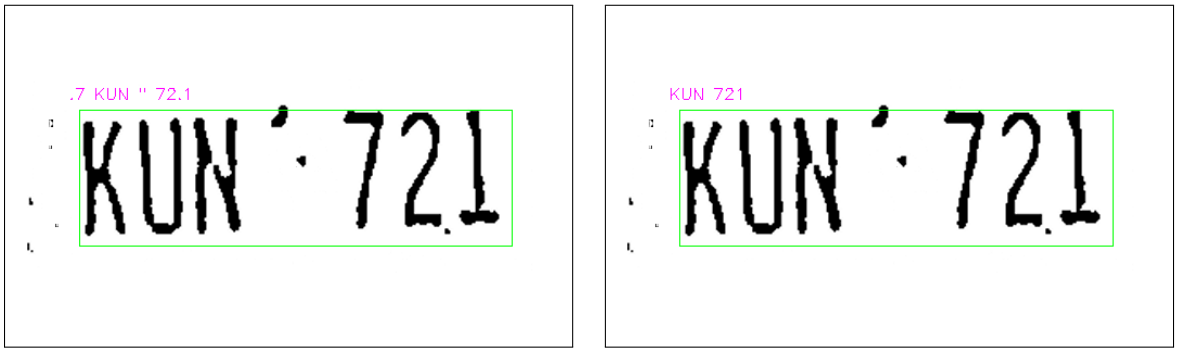


Figure 4.13: Character Recognition using Tesseract *(from left)*: using full vocabulary and limited vocabulary

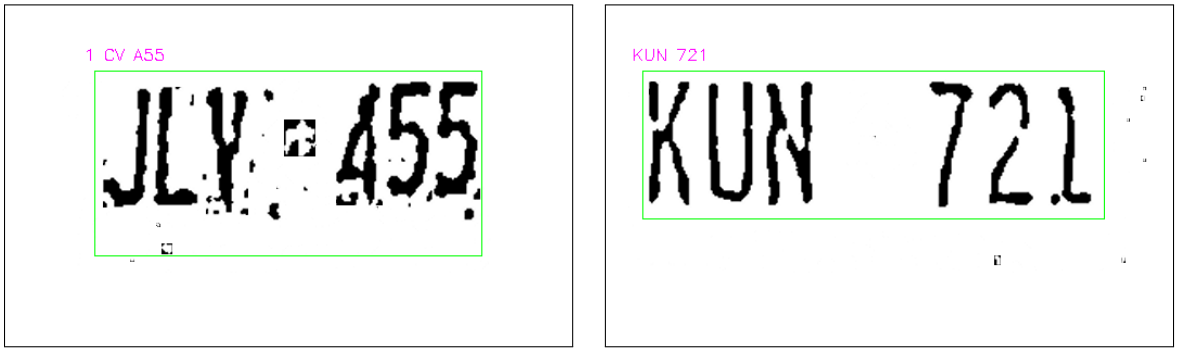


Figure 4.14: Character Recognition using Tesseract *(from left)*: Noisy binary image and clean image

# Chapter 5

# Parking lot monitoring system using the AR Drone 2.0

This chapter explains the design of the drone-based parking lot monitoring system developed in this work. The system will use the AR Drone 2.0 along with open-source software (the AR Drone SDK, OpenCV 3.0 and the Tesseract OCR engine) to fly along user-defined paths in parking lots and perform license plate recognition using video from the front camera of the drone. All components and features of the system are described.

## 5.1 Host Computer

A host computer will be used to receive navigation data from the **Navdata** service, to send control and configuration commands and to receive the video data for processing (See Section 3.5). The host computer used in this work was a Macbook Pro with 4 GB of RAM and a 2.6 GHz Intel Core 2 Duo processor running the Ubuntu 14.04 operating system. The host computer is connected to the drone via an ad-hoc Wi-Fi network of the IEEE 802.11 b/g standard.

As mentioned in Section 3.6, the motor control system of the AR Drone cannot be accessed by the user. However, the AR Drone SDK provides the application programming interface

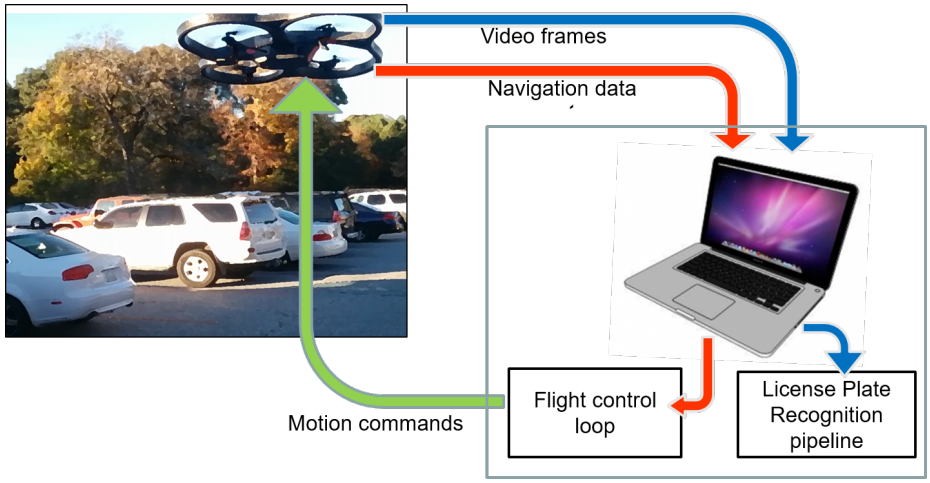


Figure 5.1: System design for drone-based license plate recognition in parking lots

to indirectly control the behavior of the drone and design custom flight paths. The flight control loop (See Section 5.3) was written using C API provided by the SDK. The computer vision algorithms in the license plate recognition pipeline (Chapter 4) were written implemented using the C++ API provided by OpenCV.

The design of the drone-based parking lot monitoring system is shown in Figure 5.1. Navigation data and video are received by the host computer from the drone. The host computer runs two processes: the flight control loop and the license plate recognition pipeline. Video frames are analyzed by the license plate recognition pipeline to recognize plates of vehicles in the parking lot. Navigation data is used by the flight control loop to determine the control commands which are then communicated to the drone.

## 5.2 Fly-Py

A new application called Fly-Py was developed using Python 2.7 in order to provide a graphical user interface for extracting GPS points from a map. Figure 5.2 shows a screenshot of the operation of Fly-Py. After specifying a location in the interface and clicking the **Go** button, an online map utility called GPS Visualizer is loaded. A custom flight path can be drawn on the map of the desired location by pressing the **Draw track**

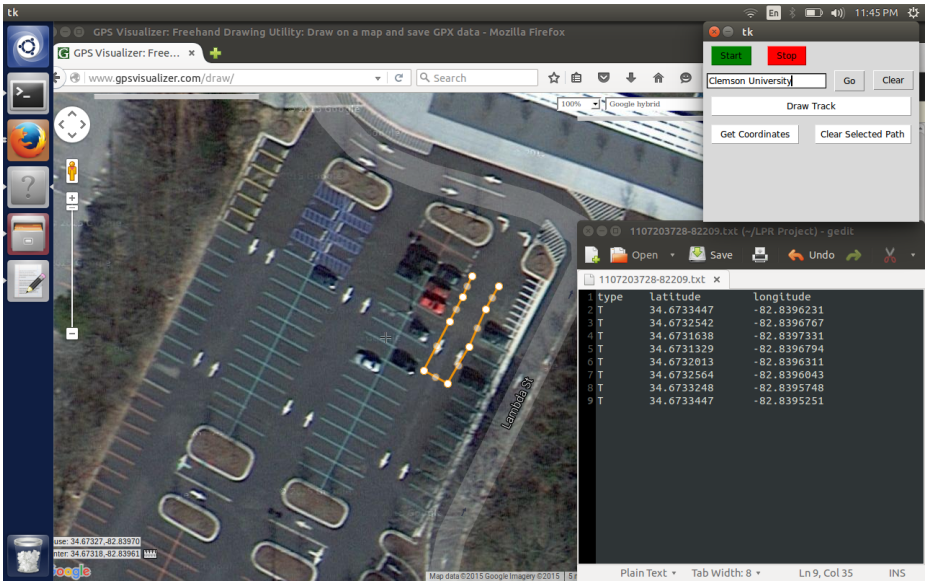


Figure 5.2: Fly-Py: a Python application to select the GPS coordinates in the parking lot for the drone flight path

button. The white circles in Figure 5.2 indicate the selected points in the parking lot while the orange line represents the flight path. GPS coordinates of the selected points can be obtained as a text file by clicking **Get coordinates**.

## 5.3 Flight Control loop

Once the text file containing the GPS points has been generated, these points are evaluated by the flight control loop to determine next set of arguments for the next **ardrone_set_progressive_cmd** command to the drone. This loop was written using the C programming language, which is supported by the AR Drone SDK. The flight control loop flies the drone from one GPS point $(lat_i, lon_i)$ to another GPS point $(lat_{i+1}, lon_{i+1})$ where $lat_i$ and $lon_i$ are the latitude and longitude (in radians) respectively. Since the GPS coordinates represent points on a sphere, equirectangular projection is used to map them into 2D space. The desired heading $\Psi_r$ is determined using $x$ and $y$ as seen in Equation 5.1.

$$x = \cos(lat_i)\sin(lat_{i+1}) - \sin(lat_i).\cos(lat_{i+1}).\cos(lon_{i+1} - lon_i) \tag{5.1}$$

$$y = \sin(lon_{i+1} - lon_i)\cos(lat_{i+1}) \tag{5.2}$$

$$\Psi_r = \frac{180}{\pi}(\tanh 2(\frac{y}{x})) \tag{5.3}$$

The Haversine formula is used to calculate shortest distance between two points on a sphere, as measured along the surface of the sphere. Assuming the earth to be a sphere of radius $R = 6378137$ metres, the following equations can be used to estimate the distance $d$ between two GPS coordinates in metres $(m)$,

$$\Delta lon = lon_2 - lon_1 \tag{5.4}$$

$$\Delta lat = lat_2 - lat_1 \tag{5.5}$$

$$a = \sin^2(\Delta lat/2) + \cos(lat_1)\cos(lat_2)\sin^2(\Delta lon/2) \tag{5.6}$$

$$c = 2\tanh\left(\sqrt{\frac{a}{1-a}}\right) \tag{5.7}$$

$$d = R \cdot c \tag{5.8}$$

The current heading of the drone $(\Psi)$ can be obtained from the **Navdata** service (see Section 3.5). Therefore, the heading error can be computed as,

$$\Delta\Psi = \Psi - \Psi_r \tag{5.9}$$

As mentioned in Section 3.6, the SDK allows the user to set a maximum value for $\Phi$ and $\Theta$ through using the **ardrone_tool** service in the application programming interface. This not only places an upper bound on the drone velocities $V_x$ and $V_y$ but also ensures that the recorded images are not rotated significantly, thereby allowing license plate recognition. For this system, the maximum roll $(\Phi)$ and pitch $(\Theta)$ angles were set to 7 degrees. The maximum flying altitude was set to 3.5 metres.

The drone velocities $(V_x, V_y)$ along the $x$ and $y$ axes can be controlled by specifying the values for pitch $(\Theta)$ and roll $(\Phi)$ as a fraction of preset maximum values. The heading of the drone is controlled by specifying yaw speed $(\Psi)$. The altitude can be controlled by setting the vertical speed $(V_z)$.

The flight control loop implemented for this system consists of two proportional gain controllers $C_1$ and $C_2$:

- In controller $C_1$, a reference velocity $V_{ref}$ is computed as a function of the distance $d$ with a gain of 0.2.

$$V_{ref} = 0.2\,d \tag{5.10}$$

  Then, the reference velocity is separated into the components along the $x$ and $y$ axes.

$$V_{rx} = V_{ref} \cos\big(\frac{\pi \Delta\Psi}{180}\big) \tag{5.11}$$

$$V_{ry} = V_{ref} \sin\big(\frac{\pi \Delta\Psi}{180}\big) \tag{5.12}$$

  Comparing $V_{rx}$ and $V_{ry}$ with current velocities $V_x$ and $V_y$ obtained from the **navdata_demo** struct (see Section 3.5), the values of $\Theta$ and $\Phi$ are estimated as shown below.

$$\Theta = -0.2(V_{rx} - V_x) \tag{5.13}$$

$$\Phi = 0.2(V_{yx} - V_y) \tag{5.14}$$

- In the second controller $C_2$, yaw speed is computed using the value of the heading error $\Delta\Psi$ from Equation 5.9 and a proportional gain of 0.011.

$$\dot{\Psi} = 0.011\,\Delta\Psi \tag{5.15}$$

The **ardrone_tool** service (see Section 3.5) is then used to communicate the values for pitch ($\Theta$), roll ($\Phi$), and yaw speed ($\dot{\Psi}$). If the magnitude of $\Theta$, $\Phi$ or $\dot{\Psi}$ is greater than 1, a value of -1 or 1 is used in the corresponding argument, depending on the desired direction of motion.

## 5.4 License plate recognition

The host computer performs the task of vehicle identification using the video frames received and decoded by the **Video** thread of the application programming interface. Frames are processed by the license plate recognition pipeline (see Chapter 4) and a text file containing the plate recognition results is generated. A flow diagram of the license plate recognition pipeline is shown in Figure 5.3

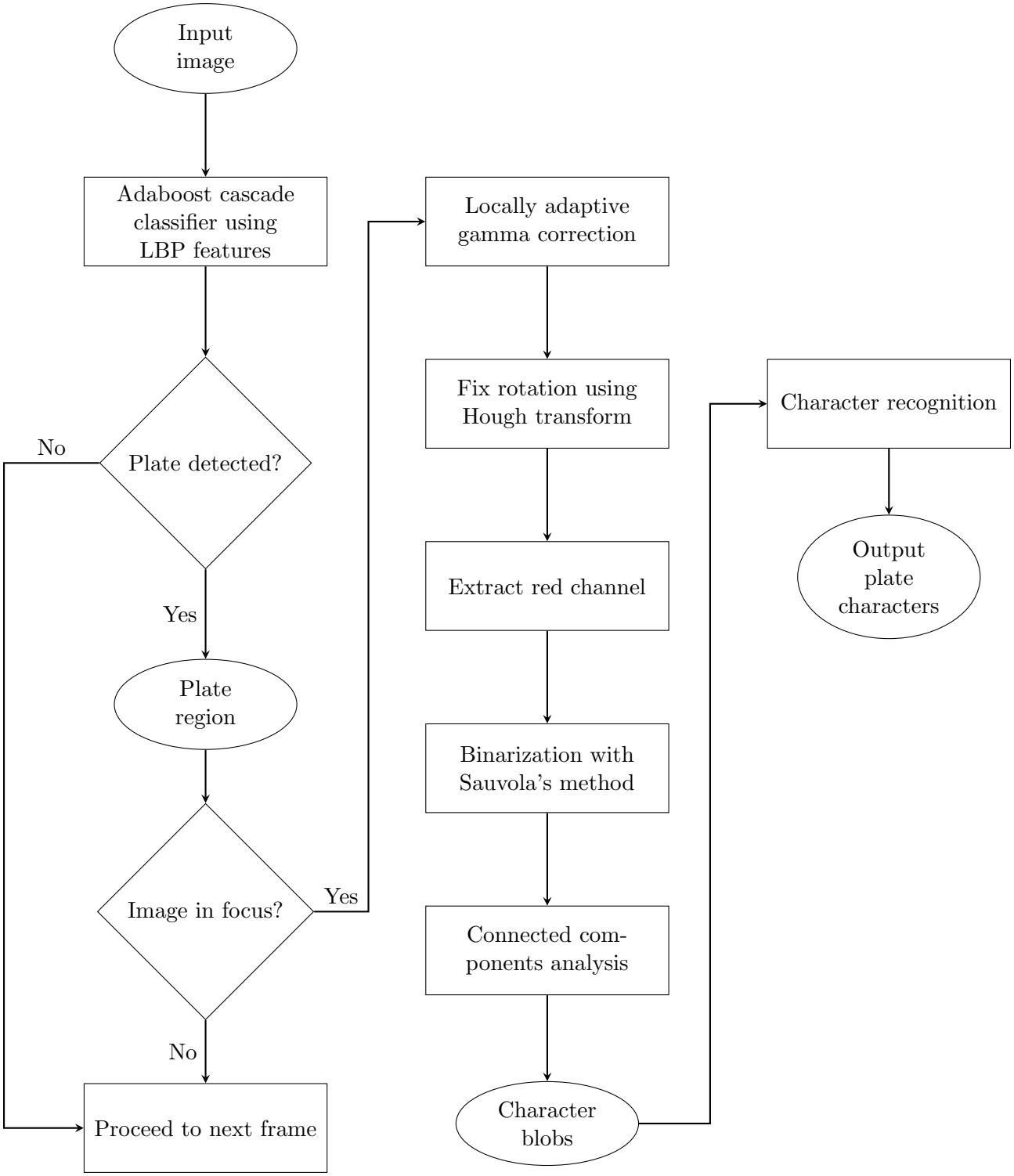


Figure 5.3: The license plate recognition pipeline

# Chapter 6

# Experiments, Results and Discussion

This chapter presents a detailed discussion of the training and testing process for the drone-based parking lot monitoring system. In order to validate its efficiency and modularity, the license plate recognition pipeline was tested with Caltech dataset in addition to drone video. Empirical analysis of experimental results are presented. Source code for the license plate recognition pipeline and the flight control loop are available online at the following Git repository:

**https://github.com/v4ven27/ar-drone**

## 6.1 Data collection and Training

Video footage was captured by flying the tethered AR Drone in 6 parking lots across Clemson, South Carolina. License plates were extracted from video frames by cropping manually. A small portion of the boundary was also included in the cropped region to allow the classifier to learn better features. Cropped images were processed to remove perspective distortion. After eliminating noisy plates and redundancies, a total of 892 license plate images and 1000 negative images were used for training. It is to be noted that over 95%


Figure 6.1: Example of license plates used for training the plate detector

of the positive training images contain license plates from the states of South Carolina, Georgia and North Carolina. Few of the positive training images are shown in Figure 6.1. The negative training set included images of trees and signs and some vehicle features (tail lamps and wheels). All the training images were resized to 40 x 25 in order to speed up the training process. The machine learning module in OpenCV automatically augments the training set by applying perspective distortions to the existing set of training images. About 4500 images were obtained by applying random distortion with maximum magnitude (in radians) of $\alpha_x = 0.1$, $\alpha_y = 0.1$ and $\alpha_z = 0.1$ along the $x, y$ and $z$ axes respectively. The XML file generated at the end of classifier training is used for plate detection in the license plate recognition pipeline.

## 6.2 The Caltech dataset

The LP recognition pipeline was tested using a common benchmark: the Caltech LP dataset. Several images from the Caltech LP dataset were added to the original training set in order to test using California license plates. It is to be noted that the images from


Figure 6.2: Caltech dataset license plate detection results

the Caltech dataset are of much better quality as compared to the drone images due to the absence of skew and blur.

Since 105 images from the Caltech dataset (out of a total 126 images) were used to train a new classifier, only 21 plate images were available for testing. Ten scene images were also used, resulting in a test set of size 31. The cascade classifier was tuned to detect plates having sizes in the range 80 x 40 and 160 x 120. Figure 6.2 shows a few plate detection results. Precision and recall were used as evaluation metrics. For the LP detection problem, precision and recall are defined as follows:

$$\text{Precision} = \frac{\text{No. of detected regions which are plates}}{\text{Total number of detected regions}} \tag{6.1}$$

$$\text{Recall} = \frac{\text{No. of plates detected}}{\text{Total number of plates}} \tag{6.2}$$

Table 6.1 shows the difference in performance between a 14-layer LBP cascade classifier with 60 features and a 17-layer cascade classifier with 100 features. It was observed that the detection rate can be increased significantly by allowing the classifier to learn more features.

The outputs of the license plate recognition pipeline for some of the Caltech images are shown in Figure 6.3. Excellent results were achieved for plate detection and character

segmentation as a direct result of the high quality of the images (Table 6.2). The 17-stage plate detector was able to identify every plate in the test set without any false positives. Character segmentation rate was observed to be 97.2% due to the loss of some characters closer to the license plate edges. The character recognition stage was evaluated using the number of segmented characters identified correctly. Due to Tesseract's limited OCR performance on scene text, the plate identification accuracy was not as high as expected.

## 6.3 Flight experiments

In order to test the license plate recognition system, the drone was programmed to fly along a user-defined path parallel to the rear end of parked vehicles, at an altitude of 3-3.5 m. The flight path was constructed by selecting GPS coordinates of several points in the parking lot using a custom Python application called Fly-Py on the host computer (see Section 5.2). This application generates GPS coordinates of the user-defined flight path as a text file. These coordinates were evaluated by the flight control loop (see Section 5.3) and corresponding motion commands were relayed to the AR Drone using an ad-hoc Wi-Fi network.

Video data was streamed from the drone to the host computer at 15 frames per second with a bitrate of 500 bits per second and resolution of 1280 x 720. The performance of the pipeline was tested by varying the drone speed between 0.75 m/s and 1.25 m/s. The effect of the distance between the drone and the vehicle ($d_v$) on plate recognition was also evaluated by adjusting $d_v$ between 3 m to 4 m.

| Number of Features | Precision | Recall |
|---|---|---|
| 60 | 0.9 | 0.86 |
| 100 | 0.95 | 1 |

Table 6.1: Performance of the 60-feature and 100-feature classifiers on the Caltech dataset

| Dataset | Caltech |
|---|---|
| Number of license plates | 21 |
| Number of plates detected | 21 |
| Number of characters in plates | 145 |
| Number of characters segmented | 141 |
| Characters identified correctly | 128 |
| Plate detection accuracy | 100% |
| Character segmentation rate | 97.2% |
| Character recognition accuracy | 90.8% |

Table 6.2: Performance of the 3 stages in license plate recognition pipeline for the Caltech dataset

### 6.3.1 Effect of drone speed on plate recognition

A non-distorted plate image is considered to be a frame containing a detected plate that is sufficiently free of blur and other noisy artifacts that can be used for character segmentation and character recognition. For this study, frames were evaluated manually in order to obtain non-distorted frames. Table 6.3 shows the performance of the plate detection stage for drone speeds between 0.75-1.25 m/s. A total of 46 plates were evaluated and the distance was fixed at $d_t = 3$ m .

It can be seen that between 0.75- 1 m/s, about half of the detected plate frames are discarded by the blur filter. However, the rejection rate is over 70% at 1.25 m/s. It was observed that increasing the drone speed resulted in fewer plate frames suitable for the license plate recognition pipeline (an average of 10 good frames per plate at 1.25 m/s as compared to 17 frames at 0.75 m/s). This phenomenon can be attributed to a number of factors. Firstly, several images are rejected by the license plate recognition pipeline because of motion blur


Figure 6.3: License plate segmentation and detection using images from the Cal Tech dataset

| Drone speed (in m/s) | 0.75 | 1.0 | 1.25 |
|---|---|---|---|
| Number of plate frames | 723 | 494 | 336 |
| Number of detected plate frames | 721 | 490 | 328 |
| Number of plate frames after blur filter | 373 | 198 | 97 |
| Frame loss (in %) | 48.3 | 59.6 | 70.5 |

Table 6.3: Table showing the relationship between drone speed and number of frames containing a single license plate

and blocking artifacts due to video streaming errors. The video processing pipeline of the AR Drone is another factor contributing to frame loss. The latency reduction mechanism used by the SDK operates such that only the most recent decodable frame in the video stream is evaluated while some older frames are dropped from the frame buffer [112]. It was also observed that more frames were dropped at higher bitrates. In order to optimize the performance of the license plate recognition system and the time of travel of the drone, the drone speed was set to 1 m/s.

### 6.3.2 Effect of distance between the drone and the vehicle on plate recognition

By considering the flying altitude of the drone and its distance from the vehicle ($d_v$), reasonable assumptions can be made about potential plate locations and dimensions. License plates are most likely to be found in the central portion of video frames. Therefore, the search space for license plate detection was limited to a small area in the middle of the image (0.25-0.70 image width and 0.40-0.95 image height), as shown in figure 6.4. For $d_v$ between 3 to 4 m, the plate detector was constrained to only locate plates having dimensions between 90 x 50 and 160 x 90.

Figure 6.4: Full video frame (*left*) and constrained search space for $d_v = 3$ m (*right*)

| Distance $d_v$ $(m)$ | 3 | 3.5 | 4 |
|---|---|---|---|
| Total number of plate frames | 1553 | 1058 | 852 |
| Number of plate detections | 1539 | 1044 | 837 |
| Hit rate(%) | 99.1 | 98.6 | 98.2 |

Table 6.4: Effect of distance between vehicle and the drone on plate detection

Table 6.4 shows the performance of the plate detection stage for $d_v$ between 3 to 4 m. Hit rate is defined as the percentage of plates detected in all frames containing plates. This takes into account multiple instances of the same plate. The plate detector could successfully locate some distorted plates, however these plates were unsuitable for character segmentation and recognition.

Plate detection accuracy is defined as the percentage of plates detected from the total number of distinct plates. From a total of 107 distinct license plates evaluated over various distances, 103 plates were detected giving an overall plate detection accuracy of 96.26%. As can be seen from Table 6.5, plate detection accuracy decreases slightly with increase in the distance between the drone and the vehicle. It was observed that some plates are misclassified as non-plates since similar plates were not learned by the Adaboost classifier during training. Few examples for the successful detection of both SC and non-SC plates are shown in Figure 6.5.

hIn order to construct the fifth row of Table 6.5, the output of character segmentation stage was examined manually to enumerate the maximum number of characters

| Distance (in m) | 3 | 3.5 | 4 | Total |
|---|---|---|---|---|
| Number of license plates | 46 | 40 | 21 | 107 |
| Number of plates detected | 45 | 39 | 19 | 103 |
| Number of plate characters | 262 | 229 | 102 | 593 |
| Sum of maximum characters segmented per plate | 237 | 212 | 78 | 527 |
| Characters identified correctly | 206 | 170 | 59 | 435 |
| Plate detection accuracy (in %) | 97.8 | 97.5 | 90.48 | 96.26 |
| Character segmentation rate (in %) | 90.46 | 92.58 | 76.47 | 88.87 |
| Character recognition accuracy (in %) | 86.91 | 80.18 | 75.64 | 82.54 |

Table 6.5: Performance of the three stages of the license plate recognition pipeline for distance $d_v$ between 3 to 4 m

segmented for each plate. Characters were deemed to be properly segmented if there was a high likelihood of them being classified correctly at the character recognition stage, taking into consideration factors like clarity, inter-character separation and presence of noise. Figure 6.6 shows some examples for poor character segmentation. It was observed that the character segmentation rate suffered with increase in distance, dropping from 90.46% at 3 m to 76.47% at 4 m (Table 6.5). This is most likely a consequence of the decrease in size of the detected license plates, which leads to very small character size. For a plate detected at 4 m, each character was about 7 pixels wide as compared to 14 pixels and 11 pixels at 3 m and 3.5 respectively. Furthermore, smaller plates are more sensitive to noise, blur and other artifacts. These adverse effects are amplified when the image is resized to improve inter-character separation in the character segmentation stage (see Section 4.3.2).


Figure 6.5: License plate detection in drone video frames

An OCR engine such as Tesseract relies only on the binary input image. Therefore, the use of the poorly segmented characters will result in sub-optimal OCR results. The scope of this thesis is to demonstrate the idea that the Tesseract OCR engine can be used to satisfactorily read license plates given optimally segmented characters. Therefore, for the purpose of this study, the accuracy of character recognition is tested using plate images that have at least 80% of characters segmented correctly. For instance, SC plates containing at least 5 segmented characters were considered. Bad segmentation results were discarded manually. Figure 6.7 shows a few examples of character recognition on good quality binarized images. It is to be noted that **X** is incorrectly identified as **A** in the final image. Several such errors were found during testing, resulting in an average character recognition rate of 82.54%.

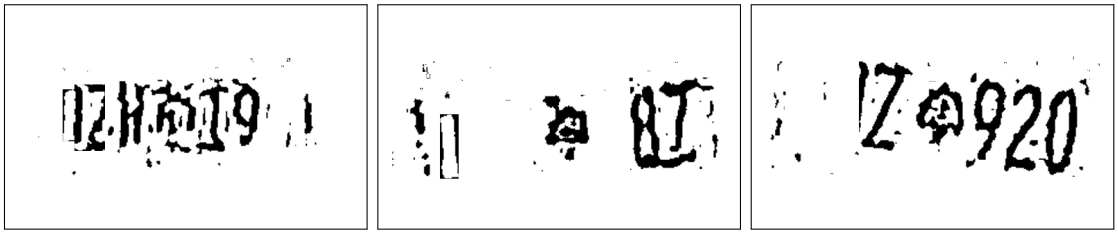

Figure 6.6: Examples of plate images with subpar character segmentation which were discarded before character recognition

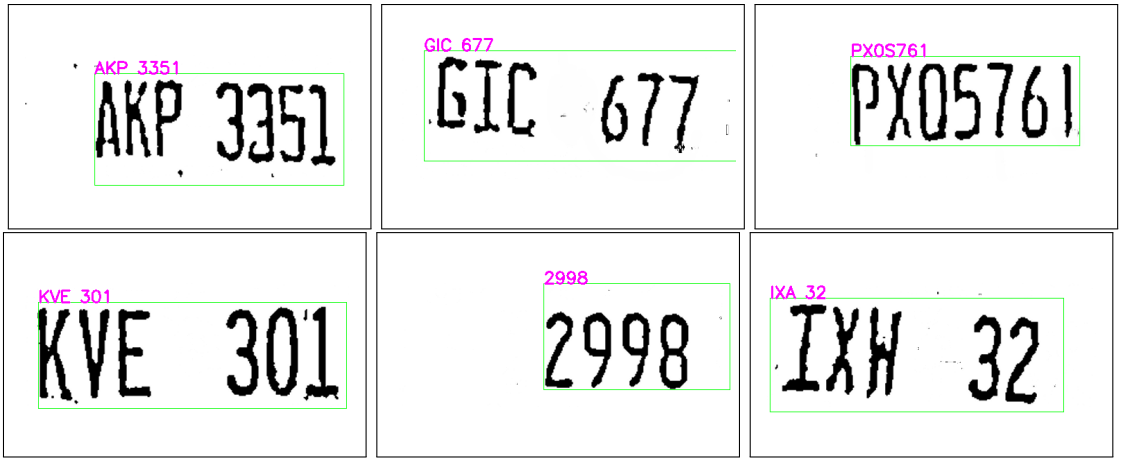


Figure 6.7: Character recognition of license plates detection the drone video using OpenCV-Tesseract

Tesseract is prone to such errors due to the feature extraction technique used in its engine (see Section 4.4). Characters with similar polynomial approximations are misclassified.

The variation of accuracy rates of the three stages are shown in Figure 6.8. Given the accuracy rates of plate detection, character segmentation and character recognition are $A_{pd}$, $A_{cs}$ and $A_{cr}$, the overall accuracy of the license plate recognition system is given by,

$$\text{System Accuracy} = A_{pd} \cdot A_{cs} \cdot A_{cr} \tag{6.3}$$

The overall accuracy of the system lies between 73-77% for distances between 3 to 3.5 m. However, the accuracy drops to about 53% at 4 m. This is mainly due to the significant drop in segmentation accuracy at 4m.

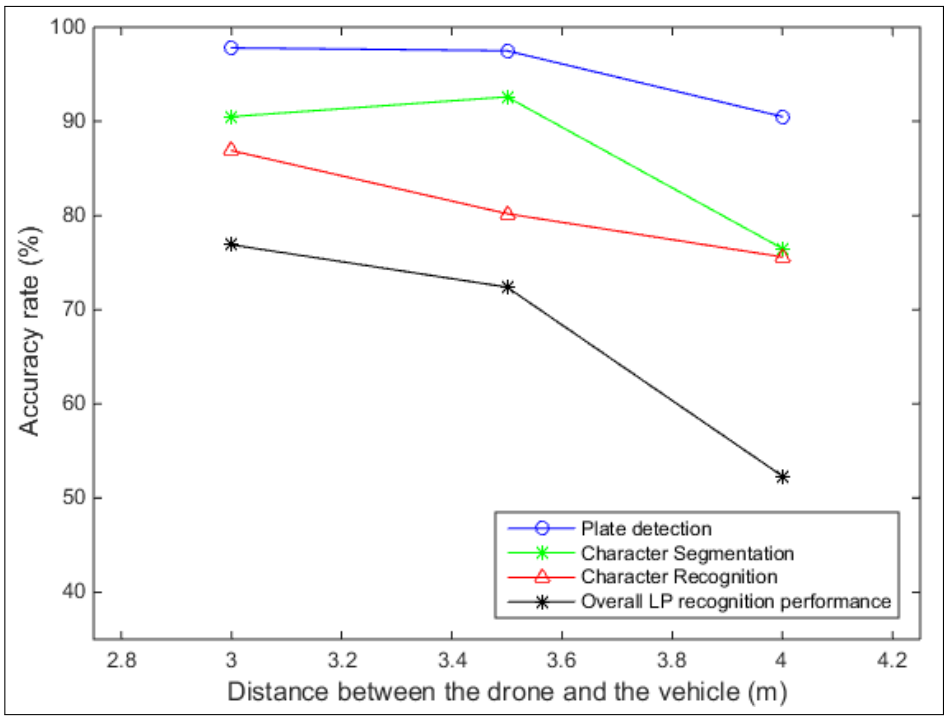


Figure 6.8: Performance of the plate detection, character segmentation and character recognition stages for distance $d_v$ between 3 to 4 m

# Chapter 7

# Conclusion and Future Work

Inspired by growing interest in applications for unmanned aerial vehicles, this work has presented a drone-based parking lot monitoring system with the ability to recognize vehicle license plates. This system differs from existing license plate recognition systems by the use of cost-efficient hardware and open source software. A drone-based system combines the non-intrusive nature of wall-mounted surveillance cameras with the mobility of patrol vehicles.

The Parrot AR Drone 2.0 was used to build and test the system due to its modularity and low cost. A custom flight control loop for the drone was implemented using the application programming interface provide by the AR Drone SDK. OpenCV 3.0 and the Tesseract OCR engine were used to implement the computer vision algorithms necessary for license plate recognition. License plates were detected in video frames at an average accuracy rate of 98.6%, validating the choice of Adaboost cascade classifiers and local binary pattern features for license plate detection. Plates were detected by the cascade classifier in approximately 50ms. Segmentation rate of 91.5% was achieved using Sauvola's method for the $d_v$ between 3 to 3.5 m. Tesseract, an engine originally meant for OCR of scanned documents, exhibited an average character recognition rate of 82.54%. The average time spent processing a frame containing a plate was about 250 ms. Therefore, the license plate recognition pipeline can be considered to have real-time performance. The overall system accuracy was 73 - 77% (for

$d_v$= 3-3.5 m). Although the performance is low compared to commercial systems, the results of the various experiments prove the feasibility of using a quadrotor drone for license plate recognition. With further research and development, drone-based systems could emulate the performance of commercial license plate recognition systems.

The greatest challenge faced during the development of the system was the quality of the video from the AR Drone 2.0. Although the video is suitable for image operations such as feature extraction, it does not appear to have sufficient quality for character segmentation. The video stream suffers from latency and blocking artifacts errors when the drone is in motion as a consequence of low bitrate and low quality camera sensor. Therefore, a number of frames are rendered unfit for license plate recognition, limiting the efficiency of the system. It might be possible to implement license plate detection using the video processor on board the AR Drone 2.0. This can potentially improve the quality of the video stream as Only detected plates are streamed to the host computer instead of the entire scene.

This system is built on the assumption that vehicles are parked such that a license plate can be observed by a passing drone. Since many US states do not require vehicles to have license plates at both the front and rear ends, license plates may be hidden from the drone camera.

Navigation accuracy of the drone was limited since the AR Drone GPS has an accuracy of $\pm 2$ m. GPS errors increase in the presence of trees or buildings due to the very nature of GPS itself. Therefore, there is a need to implement a Kalman filter for better navigation. Another solution would be to use differential GPS.

The system is also limited by the drone's Wi-Fi range of 25 m. However, the range can be increased to 50 m using a Wi-Fi range extender. The use of an ad-hoc Wi-Fi connection prevents control of multiple drones using a single host computer.

Therefore, there is plenty of room for improvement of the system developed in this work. Given that the AR Drone 2.0 model is over 3 years old, the performance of the parking lot

monitoring system can be expected to improve significantly by using a newer drone. The new Parrot Bebop drone has vastly superior flight control and video capabilities at a cost of $500 as compared to $400 for the AR Drone 2.0 with a GPS unit.

Navigation accuracy can be improved in two ways. An extended Kalman filter can be used to fuse the information from the inertial measurement unit with the GPS data to improve localization and motion. Another option would be to place tracks or guide lines along the parking lot. Using video from the vertical camera, a line-following algorithm can enable drone flight along the designated track.

A Robotic Operating System (ROS) node called ardrone_autonomy can be used to connect an AR Drone to a router-based infrastructure network. Using a host computer and multiple drones to connected to an infrastructure wireless network, a parking lot surveillance network could be developed where each drone essentially functions as a surveillance camera. Each drone could be provided with an enclosed dock or "nest" from where it performs surveillance when not in flight. Simultaneous license plate recognition in multiple parking lots can be achieved by synchronizing the flight of different drones. An automatic charging station can be provided at each "nest", resulting in a fully autonomous drone network for parking-lot monitoring.

# Bibliography


[1] John Voelcker. Car report, 2014. http://www.greencarreports.com/news/1093560_1-2-billion-vehicles-on-worlds-roads-now-2-billion-by-2035-report.

[2] This is the year drones get serious, 2015. http://fortune.com/2015/04/16/consumer-drones-2015/.

[3] Amazon consumer drones, 2013. http://www.theguardian.com/technology/2013/dec/02/amazon-consumer-drones-10-things-to-know.

[4] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[5] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[6] Ray Smith. An overview of the tesseract ocr engine. In *icdar*, pages 629–633. IEEE, 2007.

[7] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[8] Abby FineReader, 2015. http://www.abbyy.com/finereader/.

[9] Tesseract OCR, 2015. https://code.google.com/p/tesseract-ocr/.

[10] Fernando Martin, Maite Garcia, and José Luis Alba. New methods for automatic reading of vlps (vehicle license plates). In *Proc. IASTED Int. Conf. SPPRA*, pages 126–131, 2002.

[11] Bai Hongliang and Liu Changping. A hybrid license plate extraction method based on edge statistics and morphology. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 831–834. IEEE, 2004.

[12] Danian Zheng, Yannan Zhao, and Jiaxin Wang. An efficient method of license plate location. *Pattern Recognition Letters*, 26(15):2431–2438, 2005.

[13] SherrZheng Wang and Hsi-Jian Lee. Detection and recognition of license plate characters with different appearances. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 2, pages 979–984. IEEE, 2003.

[14] Jing-Ming Guo and Yun-Fu Liu. License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques. *Vehicular Technology, IEEE Transactions on*, 57(3):1417–1424, 2008.

[15] Christos-Nikolaos E Anagnostopoulos, Ioannis E Anagnostopoulos, Ioannis D Psoroulas, Vassili Loumos, and Eleftherios Kayafas. License plate recognition from still images and video sequences: A survey. *Intelligent Transportation Systems, IEEE Transactions on*, 9(3):377–391, 2008.

[16] Shan Du, Mohammad Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(2):311–325, 2013.

[17] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[18] Xiangrong Chen and Alan L Yuille. Detecting and reading text in natural scenes. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–366. IEEE, 2004.

[19] Louka Dlagnekov. License plate detection using adaboost. *Computer Science and Engineering Department, San Diego*, 2004.

[20] Clemens Arth, Florian Limberger, and Horst Bischof. Real-time license plate recognition on an embedded dsp-platform. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[21] Kuan Zheng, Yuanxing Zhao, Jing Gu, and Qingmao Hu. License plate detection using haar-like features and histogram of oriented gradients. In *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*, pages 1502–1505. IEEE, 2012.

[22] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[23] Thanh-Trung Nguyen and Thinh T Nguyen. A real time license plate detection system based on boosting learning algorithm. In *Image and Signal Processing (CISP), 2012 5th International Congress on*, pages 819–823. IEEE, 2012.

[24] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[25] ZhengHao Dong and Xin Feng. Research on license plate recognition algorithm based on support vector machine. *Journal of Multimedia*, 9(2):253–260, 2014.

[26] Xin Li. *Vehicle license plate detection and recognition.* PhD thesis, University of Missouri–Columbia, 2010.

[27] Yasuo Oguichi, Higashikubo Masakatsu, Sakda Panwai, and Ekarin Luengavilai. Automatic license plate detection and recognition in thailand. *SEI Technical Review*, (78):39, 2014.

[28] KC Kim, HR Byun, YJ Song, Young-Woo Choi, SY Chi, Kye Kyung Kim, and YunKoo Chung. Scene text extraction in natural scene images using hierarchical feature combining and verification. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 679–682. IEEE, 2004.

[29] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[30] Jong-Bae Kim. Mser and svm-based vehicle license plate detection and recognition system. In *Convergence and Hybrid Information Technology*, pages 529–535. Springer, 2012.

[31] Hao Wooi Lim and Yong Haur Tay. Detection of license plate characters in natural scene with mser and sift unigram classifier. In *Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2010 IEEE Conference on*, pages 95–98. IEEE, 2010.

[32] Jiri Matas and Karel Zimmermann. Unconstrained licence plate and text localization and recognition. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 225–230. IEEE, 2005.

[33] Bo Li, Bin Tian, Ye Li, and Ding Wen. Component-based license plate detection using conditional random field model. *Intelligent Transportation Systems, IEEE Transactions on*, 14(4):1690–1699, 2013.

[34] KV Jobin, CV Jiji, and PR Anurenjan. Automatic number plate recognition system using modified stroke width transform. In *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013 Fourth National Conference on*, pages 1–4. IEEE, 2013.

[35] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2963–2970. IEEE, 2010.

[36] Wengang Zhou, Houqiang Li, Yijuan Lu, and Qi Tian. Principal visual word discovery for automatic license plate detection. *Image Processing, IEEE Transactions on*, 21(9):4269–4279, 2012.

[37] Ching-Tang Hsieh, Yu-Shan Juan, and Kuo-Ming Hung. Multiple license plate detection for complex background. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 389–392. IEEE, 2005.

[38] Bivind Due Trier and Anil K Jain. Goal-directed evaluation of binarization methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(12):1191–1201, 1995.

[39] Lukas Neumann and Jiri Matas. A method for text localization and recognition in real-world images. In *Computer Vision–ACCV 2010*, pages 770–783. Springer, 2011.

[40] Eun Ryung Lee, Pyeoung Kee Kim, and Hang Joon Kim. Automatic recognition of a car license plate using color image processing. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 2, pages 301–305. IEEE, 1994.

[41] Xiangjian He, Lihong Zheng, Qiang Wu, Wenjing Jia, Bijan Samali, and Marimuthu Palaniswami. Segmentation of characters on car license plates. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 399–402. IEEE, 2008.

[42] Feng Yang, Zheng Ma, and Mei Xie. A novel approach for license plate character segmentation. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–6. IEEE, 2006.

[43] Shigueo Nomura, Keiji Yamanaka, Osamu Katai, Hiroshi Kawakami, and Takayuki Shiose. A novel adaptive morphological approach for degraded character image segmentation. *Pattern Recognition*, 38(11):1961–1975, 2005.

[44] Nobuo Ezaki, Marius Bulacu, and Lambert Schomaker. Text detection from natural scene images: towards a system for visually impaired persons. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 683–686. IEEE, 2004.

[45] Kwang-Baek Kim, Si-Woong Jang, and Cheol-Ki Kim. Recognition of car license plate by using dynamical thresholding method and enhanced neural networks. In *Computer Analysis of Images and Patterns*, pages 309–319. Springer, 2003.

[46] Christos Nikolaos E Anagnostopoulos, Ioannis E Anagnostopoulos, Vassili Loumos, and Eleftherios Kayafas. A license plate-recognition algorithm for intelligent transportation system applications. *Intelligent Transportation Systems, IEEE Transactions on*, 7(3):377–392, 2006.

[47] D-J Kang. Dynamic programming-based method for extraction of license plate numbers of speeding vehicles on the highway. *International Journal of Automotive Technology*, 10(2):205–210, 2009.

[48] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven. Photoocr: Reading text in uncontrolled conditions. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 785–792. IEEE, 2013.

[49] Rodrigo Minetto, Nicolas Thome, Matthieu Cord, Neucimar J Leite, and Jorge Stolfi. T-hog: An effective gradient-based descriptor for single line text regions. *Pattern recognition*, 46(3):1078–1090, 2013.

[50] Wayne Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, 1985.

[51] Sergey Milyaev, Olga Barinova, Tatiana Novikova, Pushmeet Kohli, and Victor Lempitsky. Image binarization for end-to-end text understanding in natural images. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 128–132. IEEE, 2013.

[52] Anand Mishra, Karteek Alahari, and CV Jawahar. An mrf model for binarization of natural scene text. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 11–16. IEEE, 2011.

[53] Zhu KF Qi, M Kimachi, Y Wu, and T Aziwa. Using adaboost to detect and segment characters from natural scenes. In *Proceedings of CBDAR, ICDAR Workshop*, 2005.

[54] B Gatos, I Pratikakis, K Kepene, and SJ Perantonis. Text detection in indoor/outdoor scene images. In *Proc. First Workshop of Camera-based Document Analysis and Recognition*, pages 127–132, 2005.

[55] Mauritius Seeger and Christopher Dance. Binarising camera images for ocr. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 54–58. IEEE, 2001.

[56] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545. IEEE, 2012.

[57] Myung-Chul Sung, Bongjin Jun, Hojin Cho, and Daijin Kim. Scene text detection with robust character candidate extraction method.

[58] Weilin Huang, Zhe Lin, Jianchao Yang, and Jue Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1241–1248. IEEE, 2013.

[59] Muhammad Fraz, M Saquib Sarfraz, and Eran A Edirisinghe. Exploiting colour information for better scene text recognition. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[60] Egyul Kim, SeongHun Lee, and JinHyung Kim. Scene text extraction using focus of mobile camera. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 166–170. IEEE, 2009.

[61] Chucai Yi and Yingli Tian. Scene text recognition in mobile applications by character descriptor and structure configuration. *Image Processing, IEEE Transactions on*, 23(7):2972–2982, 2014.

[62] International conference on document analysis and recognition, 2015. `http://2015.icdar.org/program/competitions/`.

[63] LI Ning. An implementation of ocr system based on skeleton matching. 1991.

[64] Min-Ki Kim and Young-Bin Kwon. Multi-font and multi-size character recognition based on the sampling and quantization of an unwrapped contour. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 170–174. IEEE, 1996.

[65] Mi-Ae Ko and Young-Mo Kim. License plate surveillance system using weighted template matching. In *null*, page 269. IEEE, 2003.

[66] Mi Ko and Young-Mo Kim. A simple ocr method from strong perspective view. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 235–240. IEEE, 2004.

[67] Yuk Ying Chung and Man To Wong. Handwritten character recognition by fourier descriptors and neural network. In *TENCON'97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, volume 1, pages 391–394. IEEE, 1997.

[68] Peifeng Hu, Yannan Zhao, Zehong Yang, and Jiaqin Wang. Recognition of gray character using gabor filters. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 1, pages 419–424. IEEE, 2002.

[69] Andrew J Newell and Lewis D Griffin. Natural image character recognition using oriented basic image features. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 191–196. IEEE, 2011.

[70] Andrew J Newell and Lewis D Griffin. Multiscale histogram of oriented gradient descriptors for robust character recognition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1085–1089. IEEE, 2011.

[71] Jieun Kim and Ho-sub Yoon. Graph matching method for character recognition in natural scene images: A study of character recognition in natural scene image considering visual impairments. In *Intelligent Engineering Systems (INES), 2011 15th IEEE International Conference on*, pages 347–350. IEEE, 2011.

[72] Yuqing Song, Jianqing Liu, and Dianwei Li. Graph grammar based license plate recognition. In *Intelligent Networks and Intelligent Systems (ICINIS), 2012 Fifth International Conference on*, pages 37–40. IEEE, 2012.

[73] Nallasamy Mani and Bala Srinivasan. Application of artificial neural network model for optical character recognition. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 3, pages 2517–2520. IEEE, 1997.

[74] Hadar Avi-Itzhak, Thanh Diep, Harry Garland, et al. High accuracy optical character recognition using neural networks with centroid dithering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(2):218–224, 1995.

[75] Annie France. A multi-font character recognition based on its fundamental features by artificial neural networks. 1996.

[76] Juan Manuel Ramirez, Pilar Gomez-Gil, and David Baez-Lopez. On structural adaptability of neural networks in character recognition. In *Signal Processing, 1996., 3rd International Conference on*, volume 2, pages 1481–1483. IEEE, 1996.

[77] Enyong Hu, Hui Wang, Jianhua Wang, Song Lu, and Lei Tian. Study on pattern recognition model based on principal component analysis and radius basis function neural network. In *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, volume 2, pages 388–390. IEEE, 2011.

[78] Feng Yang and Fan Yang. Character recognition using parallel bp neural network. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 1595–1599. IEEE, 2008.

[79] Suruchi G Dedgaonkar, Anjali A Chandavale, and Ashok M Sapkal. Survey of methods for character recognition. *International Journal of Engineering and Innovative Technology (IJEIT) Volume*, 1:180–189, 2012.

[80] H Erdinc Kocer and K Kursat Cevik. Artificial neural networks based vehicle license plate recognition. *Procedia Computer Science*, 3:1033–1037, 2011.

[81] Sorin Draghici. A neural network based artificial vision system for licence plate recognition. *International Journal of Neural Systems*, 8(01):113–126, 1997.

[82] Chao Gou, Kunfeng Wang, Zhongdong Yu, and Haitao Xie. License plate recognition using mser and hog based on elm. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 217–221. IEEE, 2014.

[83] Antonio Carlos Gay Thome. *SVM Classifiers-Concepts and Applications to Character Recognition*. INTECH Open Access Publisher, 2012.

[84] Tiago C Mota and Antonio Carlos G Thome. One-against-all-based multiclass svm strategies applied to vehicle plate character recognition. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 2153–2159. IEEE, 2009.

[85] S. Medeiros. Svm applied to license plate recognition. 2011.

[86] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[87] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: investigation of normalization and feature extraction techniques. *Pattern Recognition*, 37(2):265–279, 2004.

[88] Shriram Kishanrao Waghmare, AK Gulve, Vikas N Nirgude, and Nagraj P Kamble. Automatic number plate recognition (anpr) system for indian conditions using support vector machine (svm). *International Journal of Computer Science and its Applications*, 2010.

[89] Kumar Parasuraman and PS Subin. Svm based license plate recognition system. In *IEEE International Conference on Computational Intelligence and Computing Research*, 2010.

[90] Abdul Rahim Ahmad, Christian Viard-Gaudin, and Marzuki Khalid. Lexicon-based word recognition using support vector machine and hidden markov model. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 161–165. IEEE, 2009.

[91] Cui Lin and Ruan Qiuqi. A novel svm classifier for recognition of gray character using gabor filters. In *Signal Processing, 2004. Proceedings. ICSP'04. 2004 7th International Conference on*, volume 2, pages 1451–1454. IEEE, 2004.

[92] Yue Xue. Optical character recognition using template matching and svm. 2011.

[93] Chucai Yi, Xiaodong Yang, and Yingli Tian. Feature representations for scene text character recognition: A comparative study. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 907–911. IEEE, 2013.

[94] Bo Li, Bin Tian, Qingming Yao, and Kunfeng Wang. A vehicle license plate recognition system based on analysis of maximally stable extremal regions. In *Networking, Sensing and Control (ICNSC), 2012 9th IEEE International Conference on*, pages 399–404. IEEE, 2012.

[95] Guanghan Ning. *Vehicle license plate detection and recognition*. PhD thesis, University of Missouri–Columbia, 2013.

[96] Lixia Liu, Honggang Zhang, Aiping Feng, Xinxin Wan, and Jun Guo. Simplified local binary pattern descriptor for character recognition of vehicle license plate. In *Computer Graphics, Imaging and Visualization (CGIV), 2010 Seventh International Conference on*, pages 157–161. IEEE, 2010.

[97] Jayke Meijer. License plate recognition using local binary patterns, 2012.

[98] Md Mostafa Kamal Sarker, Moon Kyou Song, and Jeonbuk Iksan. Korean car license plate character recognition using local line binary pattern.

[99] Teófilo Emídio de Campos, Bodla Rakesh Babu, and Manik Varma. Character recognition in natural images. In *VISAPP (2)*, pages 273–280, 2009.

[100] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 440–445. IEEE, 2011.

[101] Tao Wang, David J Wu, Andrew Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[102] Tomáš Krajník, Vojtěch Vonásek, Daniel Fišer, and Jan Faigl. Ar-drone as a platform for robotic research and education. In *Research and Education in Robotics-EUROBOT 2011*, pages 172–186. Springer, 2011.

[103] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 5776–5783. IEEE, 2011.

[104] Takayoshi Mori and Stefan Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1750–1757. IEEE, 2013.

[105] Arnoud Visser, Nick Dijkshoorn, Martijn Van der Veen, and Robrecht Jurriaans. Closing the gap between simulation and reality in the sensor and motion models of an autonomous ar. drone. In *International Micro Air Vehicle conference and competitions 2011 (IMAV 2011),'t Harde, The Netherlands, September 12-15, 2011*. Delft University of Technology and Thales, 2011.

[106] Nick Dijkshoorn and Arnoud Visser. Integrating sensor and motion models to localize an autonomous ar. drone. *International Journal of Micro Air Vehicles*, 3(4):183–200, 2011.

[107] Alessandro Benini, Adriano Mancini, and Sauro Longhi. An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15. 4a wireless sensor network. *Journal of Intelligent & Robotic Systems*, 70(1-4):461–476, 2013.

[108] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Camera-based navigation of a low-cost quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2815–2821. IEEE, 2012.

[109] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Accurate figure flying with a quadrocopter using onboard visual and inertial sensing. *IMU*, 320:240, 2012.

[110] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11):1646–1656, 2014.

[111] Parrot. Parrot AR Drone 2.0 specifications, 2012. `http://ardrone2.parrot.com/ardrone-2/specifications/`.

[112] Parrot. AR Drone developer guide. 2013.

[113] Pierre-Jean Bristeau, François Callou, David Vissiere, Nicolas Petit, et al. The navigation and control technology inside the ar. drone micro uav. In *18th IFAC world congress*, volume 18, pages 1477–1484, 2011.

[114] Nick Dijkshoorn. Simultaneous localization and mapping with the ar. drone. *PhD diss., Masters thesis, Universiteit van Amsterdam*, 2012.

[115] Song Ke-Chen, YAN Yun-Hui, CHEN Wen-Hui, and Xu ZHANG. Research and perspective on local binary pattern. *Acta Automatica Sinica*, 39(6):730–744, 2013.

[116] Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang, and Stan Z Li. Learning multi-scale block local binary patterns for face recognition. In *Advances in Biometrics*, pages 828–837. Springer, 2007.

[117] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.

[118] Dongni Zhang, Won-Jae Park, Seung-Jun Lee, Kang-A Choi, and Sung-Jea Ko. Histogram partition based gamma correction for image contrast enhancement. In *Consumer Electronics (ISCE), 2012 IEEE 16th International Symposium on*, pages 1–4. IEEE, 2012.

[119] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[120] Chucai Yi and YingLi Tian. Text string detection from natural scenes by structure-based partition and grouping. *Image Processing, IEEE Transactions on*, 20(9):2594–2605, 2011.

[121] Chucai Yi and Yingli Tian. Text extraction from scene images by character appearance and structure modeling. *Computer Vision and Image Understanding*, 117(2):182–194, 2013.

[122] Khaoula Elagouni, Christophe Garcia, Franck Mamalet, and Pascale Sébillot. Combining multi-scale character recognition and linguistic knowledge for natural scene text ocr. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 120–124. IEEE, 2012.

[123] Rodrigo Minetto, Nicolas Thome, Matthieu Cord, Jorge Stolfi, Frédéric Precioso, Jonathan Guyomard, and Neucimar J Leite. Text detection and recognition in urban scenes. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 227–234. IEEE, 2011.